Détection des intrusions dans les systèmes d'information : la nécessaire prise en compte des caractéristiques du système surveillé

Ludovic Mé

HABILITATION À DIRIGER DES RECHERCHES

présentée devant

L'Université de Rennes 1
Institut de Formation Supérieure
en Informatique et en Communication

par

Ludovic MÉ[1]

Détection des intrusions dans les systèmes d'information : la nécessaire prise en compte des caractéristiques du système surveillé

soutenue le 16 juin 2003 devant le jury composé de

MM.	Yves Deswarte	Rapporteur
	Refik Molva	Rapporteur
	Pierre Rolin	Rapporteur
	Frédéric Cuppens	Examinateur
Mme	Mireille Duccassé	Examinateur
MM.	Yves Quenec'hdu	Examinateur
	Gerardo Rubino	Examinateur

[1]Supélec, campus de Rennes, équipe SSIR

Table des matières

Présentation du document **1**

1 Introduction **3**
1.1 Problématique de la détection d'intrusions 3
1.2 Guide de lecture 4
1.3 Une terminologie s'appuyant sur un modèle général de système de détection d'intrusions 5

2 Caractéristiques des IDS et positionnement de nos travaux **7**
2.1 Sources des données à analyser 7
2.1.1 Brève description des différentes sources 7
2.1.2 Avantages et inconvénients des différentes sources 9
2.2 Principes de détection 10
2.2.1 L'approche comportementale 10
2.2.2 L'approche par scénarios 13
2.2.3 Avantages et inconvénients des deux approches 16
2.3 Architecture 19
2.4 Granularité de l'analyse 19
2.5 Comportements en cas de détection 20
2.6 Positionnement de nos travaux 21
2.6.1 Vis-à-vis des sources de données 21
2.6.2 Vis-à-vis de la méthode de détection 21
2.6.3 Vis-à-vis de l'architecture de l'IDS 25
2.6.4 Vis-à-vis de la granularité de l'analyse 27
2.6.5 Vis-à-vis du comportement après détection 27

3 Prise en compte des caractéristiques du système surveillé et de sa politique de sécurité **29**
3.1 Motivations 29
3.2 Corrélation d'alertes avec prise en compte de l'environnement surveillé 31
3.2.1 Objectifs de l'étude 31
3.2.2 Différentes formes de corrélation 32
3.2.3 Connaissances nécessaires à la déduction 33
3.2.4 M2D2, un modèle formel pour la corrélation d'alertes 35

3.2.5 Exemples d'exploitation de M2D2 en corrélation d'alertes : identification des faux positifs 41
3.2.6 Avantages et limites 43
3.2.7 Conclusion 43
3.3 Détection d'intrusions orientée politique basée sur un modèle de contrôle de flux d'information 44
3.3.1 Systèmes d'exploitation classiques et attaques par délégation 44
3.3.2 Détecter les attaques par délégation 45
3.3.3 Cas du contrôle d'accès discrétionnaire 46
3.3.4 Limitation du nombre de faux positifs 47
3.3.5 Un modèle de contrôle de flux adapté à la détection des attaques par délégation 47
3.3.6 Exemple d'implantation d'une politique de sécurité simple par le biais du contrôle des flux de référence 52
3.3.7 Exemple de détection d'attaque 53
3.3.8 Réaction à une opération illégale : schémas de détection et de prévention 55
3.3.9 Avantages et limites 55

4 Perspectives de recherche **59**
4.1 Amélioration des sondes de détection 59
4.1.1 Approche comportementale 60
4.1.2 Approche par scénario 61
4.1.3 Prise en compte de l'environnement par les sondes 62
4.2 Amélioration des fonctions du manager 62
4.3 Evaluation des IDS 63

Bibliographie **64**

Présentation du document

A l'issue de mes études d'ingénieur, j'hésitais entre deux voies : m'engager dans une carrière industrielle au service du public, ou m'engager dans une carrière d'enseignant-chercheur. En septembre 1987, je tranchais en intégrant la direction « Distribution » d'EDF, en qualité d'ingénieur d'étude. Rapidement, je me rendais compte que ce type d'environnement de travail ne me convenait pas. Je décidais donc de me tourner vers l'enseignement et la recherche et entrais à Supélec dès mai 1988.

En 1990, Bernard Picinbono devenait Directeur de Supélec. Souhaitant développer dans les équipes propres de l'école une politique de recherche, certes originale par l'importance accordée aux relations industrielles, mais se rapprochant des critères universitaires, B. Picinbono proposait aux enseignants volontaires de s'engager dans un travail de thèse. C'est une opportunité que j'attendais et que je saisissais. En janvier 1991, je m'inscrivais en thèse, sous la direction de Pierre Rolin.

Depuis cette date, mes travaux portent sur la sécurité des systèmes d'information et plus précisément sur la détection des intrusions dans les systèmes et réseaux informatiques.

Initiés en thèse, ces travaux ont ensuite été poursuivis et amplifiés, notamment grâce à la possibilité qui m'a été offerte à Supélec en 1997, de créer et de prendre la responsabilité de l'équipe « Sécurité des Systèmes d'Information et Réseaux », qui compte aujourd'hui 7 enseignants-chercheurs et 8 thésards.

Ce mémoire propose une synthèse de ces travaux de recherche. Son introduction pose la problématique de la détection d'intrusions et définit les termes utilisés dans la suite. Le premier chapitre présente les grandes caractéristiques des systèmes de détection d'intrusions puis positionne les travaux menés relativement à ces caractéristiques. Le deuxième chapitre met en avant, parmi les travaux en cours, ceux qui paraissent les plus à même d'apporter des réponses aux problèmes actuels de la détection d'intrusions. Enfin, ce mémoire se conclue par la présentation de quelques perspectives de recherche.

Chapitre 1

Introduction

1.1 Problématique de la détection d'intrusions

L'informatique mondiale a été bouleversée depuis le début des années 90 par l'avènement de l'internet, puis sa « démocratisation » grâce au web. Les entreprises n'ont plus le choix : non connectées, elles n'auraient plus d'avenir. Le nombre d'ordinateurs personnels inter-connectés est maintenant immense. Même les administrations, sous la pression des citoyens, mais aussi par souci de « modernisme », offrent de plus en plus de services au travers de l'internet. Ainsi, depuis 2000, il est possible de télé-déclarer ses revenus sur le site du ministère des finances.

Si ce bouleversement semble profitable à tous du point de vue économique, il ne va pas sans risque. Toutes ces machines inter-connectées sont autant de cible pour les attaquants.

En effet, le « réseau des réseaux » a été conçu à une époque où l'internet servait de moyen de communication à une communauté de chercheurs en informatique. Tous travaillaient de concert pour faire fonctionner ce réseau, malgré les nombreuses défaillances des moyens de communication de l'époque. Dans un tel contexte, la confiance était la règle, la malveillance inexistante. En conséquence, il était inutile de prévoir dans les mécanismes en création des fonctions dédiées à leur sécurité. En revanche, il était important de faire en sorte que les équipements connectés offrent un maximum d'information les concernant, afin de faciliter l'opération et la mise au point du réseau. Aujourd'hui, ces mécanismes conçus dans les années 70 sont toujours utilisés, pratiquement inchangés. Il est donc facile à un attaquant d'exploiter leur ouverture, non plus au profit de la communauté, mais dans un but malicieux [Des01].

En outre, l'internet permet aujourd'hui à tout un chacun, grâce au web et à ses moteurs de recherche, d'accéder à une masse considérable d'informations relatives aux « attaques informatiques ». Sur le web, on peut trouver relativement simplement du code prêt à l'emploi (des « *exploits* ») implantant telle ou telle technique d'attaque. Il est donc de plus en plus facile de s'improviser « pirate ».

Les failles de sécurité sont nombreuses ; ceux qui sont prêts à les exploiter aussi.

Le danger est donc réel. Il l'est d'autant plus que dans un marché toujours plus concurrentiel, les fournisseurs de produits et de services mettent plus l'accent sur la réduction des coûts et l'amélioration de la qualité, que sur la sécurité. Ce constat nous paraît particulièrement frappant en ce qui concerne les fournisseurs d'accès à l'internet, que la loi du marché pousse à mettre en avant, non pas la sécurité de leurs abonnés, mais des prix attractifs et une meilleure qualité de service (débits, disponibilité, fonctionnalités, etc.).

Néanmoins, le marché de la sécurité explose. Les utilisateurs et les fournisseurs de solutions ont donc compris qu'il fallait se protéger. La sécurité est devenue un souci majeur. Nombre de mécanismes préventifs existe, qui cherche à offrir les cinq grands services de sécurité : authentification, contrôle d'accès, confidentialité, intégrité, disponibilité. Pour autant, un constat s'impose : la plupart des ordinateurs connectés à l'internet restent vulnérables aux attaques. La mise en place d'outils permettant de détecter des malversations (intrusions) est donc indispensable.

1.2 Guide de lecture

Ce mémoire résume 10 ans de travail autour de l'étude des concepts à la base des systèmes de détection d'intrusions. Il est organisé comme suit.

Le chapitre 2 présente les grandes caractéristiques des systèmes de détection d'intrusions (la source des données à analyser, l'approche de détection retenue, l'architecture de l'outil de détection d'intrusions, la granularité de l'analyse et le comportement en cas de détection), puis positionne nos travaux relativement à ces caractéristiques. Nos principales contributions sont mises en évidence dans le tableau 2.2, page 22.

Dans le chapitre 3, nous avons choisi de mettre en avant parmi nos travaux en cours, ceux qui nous parraissent les plus à même d'apporter des réponses aux problèmes actuels de la détection d'intrusions. Ces travaux présentent une caractéristique commune : ils prennent en compte les caractéristiques du système surveillé : topologie, types de machine, logiciels installés, failles connues, politique de sécurité en vigueur. Nous présentons dans un premier temps le travail fait dans le cadre de la thèse de Benjamin Morin (sous la direction de Mireille Ducassé, INSA de Rennes et la co-direction conjointe de Hervé Debar, France Télécom R&D) autour de la corrélation des alertes issues des outils de détection d'intrusions. Dans un second temps, nous présentons les travaux faits dans le cadre de la thèse de Jacob Zimmerman (sous la direction de Gerardo Rubino, Université de Rennes 1), autour du contrôle d'accès tolérant aux intrusions.

Comme il se doit, ce mémoire se conclue par la présentation de quelques perspectives de recherche.

Avant d'aborder le chapitre 2, nous présentons un modèle général de système de détection d'intrusions, sur lequel nous appuyons une terminologie. C'est l'objet du

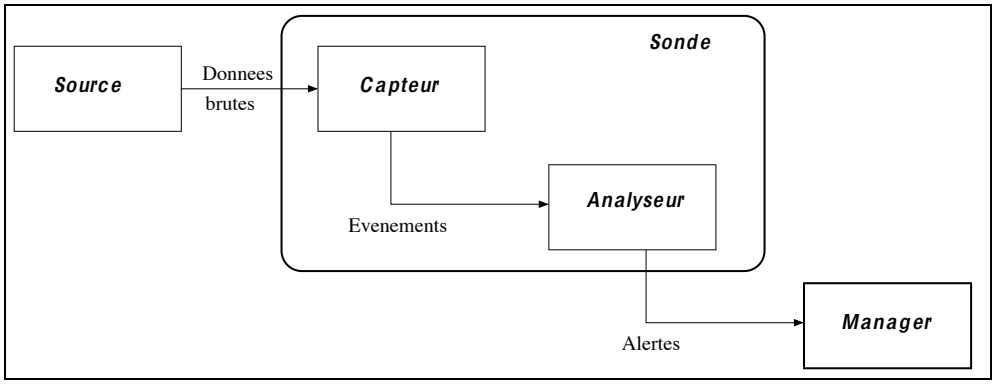


FIG. 1.1 – Le modèle général proposé par le groupe IDWG (*Intrusion Detection Working Group*) de l'IETF.

paragraphe suivant. Sa lecture est importante puisqu'on y trouve les définitions des termes spécifiques utilisés dans ce mémoire.

1.3 Une terminologie s'appuyant sur un modèle général de système de détection d'intrusions

La domaine de la détection d'intrusions est relativement jeune. C'est en 1980 seulement que J.P. Anderson en posait les prémisses dans son célèbre rapport *Computer Security Threat Monitoring and Surveillance* [And80]. Aujourd'hui encore, les membres de la communauté appellent parfois différemment des choses semblables et de manière semblable des choses différentes. Récemment, des efforts ont été faits pour définir une terminologie s'appuyant sur un modèle général de détection d'intrusions [WE02]. Je m'appuierai sur ce modèle et j'adopterai cette terminologie tout au long de ce mémoire.

Les différents éléments de ce modèle (voir figure 1.1) sont les suivants :
- **source de données** : dispositif générant de l'information sur les activités des entités du système d'information. Exemple : *sniffers* réseau, système d'audit d'un système d'exploitation, etc.
- **capteur** : dispositif génèrant des événements en filtrant et formatant les données brutes provenant d'une source de données.
- **événement** : message formaté émis par un capteur. C'est l'unité élémentaire utilisée pour représenter une étape d'un scénario d'attaque connu. Ces événements sont parfois appelés événements d'audit, ou données d'audit.
- **analyseur** : dispositif analysant les événements à la recherche de traces d'intrusions.
- **alerte** : message formaté émis par un analyseur s'il trouve des traces d'intru-

sions.
- **sonde** : ensemble constitué d'un capteur et d'un analyseur.
- **manager** : composant permettant à l'administrateur du système de configurer les différents éléments (capteur, analyseur) et de gérer les alertes reçues.

Il faut noter que le modèle de la figure 1.1 est par nature récursif. En effet, un manager peut être considéré par un capteur comme une source de données. C'est là une caractéristique très importante sur laquelle nous reviendrons lorsque nous aborderons la corrélation d'alertes.

Quelques termes n'apparaissant pas directement sur la figure 1.1 doivent aussi être définis :
- **attaque** ou **intrusion** : action intentionnelle ayant pour conséquence de compromettre l'intégrité, la confidentialité ou la disponibilité d'un système d'information. En d'autres termes, toute violation de la politique de sécurité.
- **scénario** : suite constituée des étapes élémentaires d'une attaque.
- **signature** : suite des étapes observables d'une attaque, utilisée par certains analyseurs.
- **incident de sécurité**[1] : ensemble d'attaques ou d'intrusions, éventuellement réduit à un singleton.
- **détection d'intrusions** : recherche de traces laissées par une intrusion dans les données produites par une source.
- **faux positif** : alerte émise par un analyseur en absence d'attaque (fausse alerte).
- **faux négatif** : absence d'alerte en présence d'attaque.
- **système de détection d'intrusions** : ensemble constitué d'un ou plusieurs capteurs, un ou plusieurs analyseurs et un ou plusieurs managers. Son objectif est de détecter automatiquement toute intrusion dans un système d'information. On le désignera par son acronyme anglais, IDS (*Intrusion Detection System*).
- **réaction** : mesures passives ou actives prises en réponse à la détection d'une intrusion, pour la stopper ou pour corriger ses effets.

[1] terminologie empruntée à Hervé Debar.

Chapitre 2

Caractéristiques des IDS et positionnement de nos travaux

Introduction

Il existe de nombreux systèmes de détection d'intrusions. Une centaine d'outils est répertoriée dans [Sob00], une soixantaine dans [net03]. Nous abordons dans ce chapitre leurs différentes caractéristiques (paragraphes 2.1 à 2.5). Le paragraphe 2.6 positionne quant à lui nos travaux par rapport à ces caractéristiques.

Les critères de classification adoptés ici (voir figure 2.1) sont inspirés de ceux présentés dans [DDW00] et [Axe99, Axe00] : la source des données à analyser, l'approche de détection retenue, l'architecture de l'IDS, la granularité de l'analyse et, enfin, son comportement en cas de détection.

2.1 Sources des données à analyser

Détecter une intrusion, c'est en trouver les traces dans des données qu'on analyse. La nature et la source de ces données sont donc bien sûr des caractéristiques essentielles.

2.1.1 Brève description des différentes sources

Les données peuvent provenir du système d'exploitation ou des applications qui y résident (on parle alors de HIDS, pour *Host-based IDS*).

Le système d'exploitation lui-même propose plusieurs sources d'information, telles que l'historique des commandes passées par les utilisateurs ou l'*accounting*, qui fournit de l'information sur l'usage fait par les utilisateurs des ressources partagées

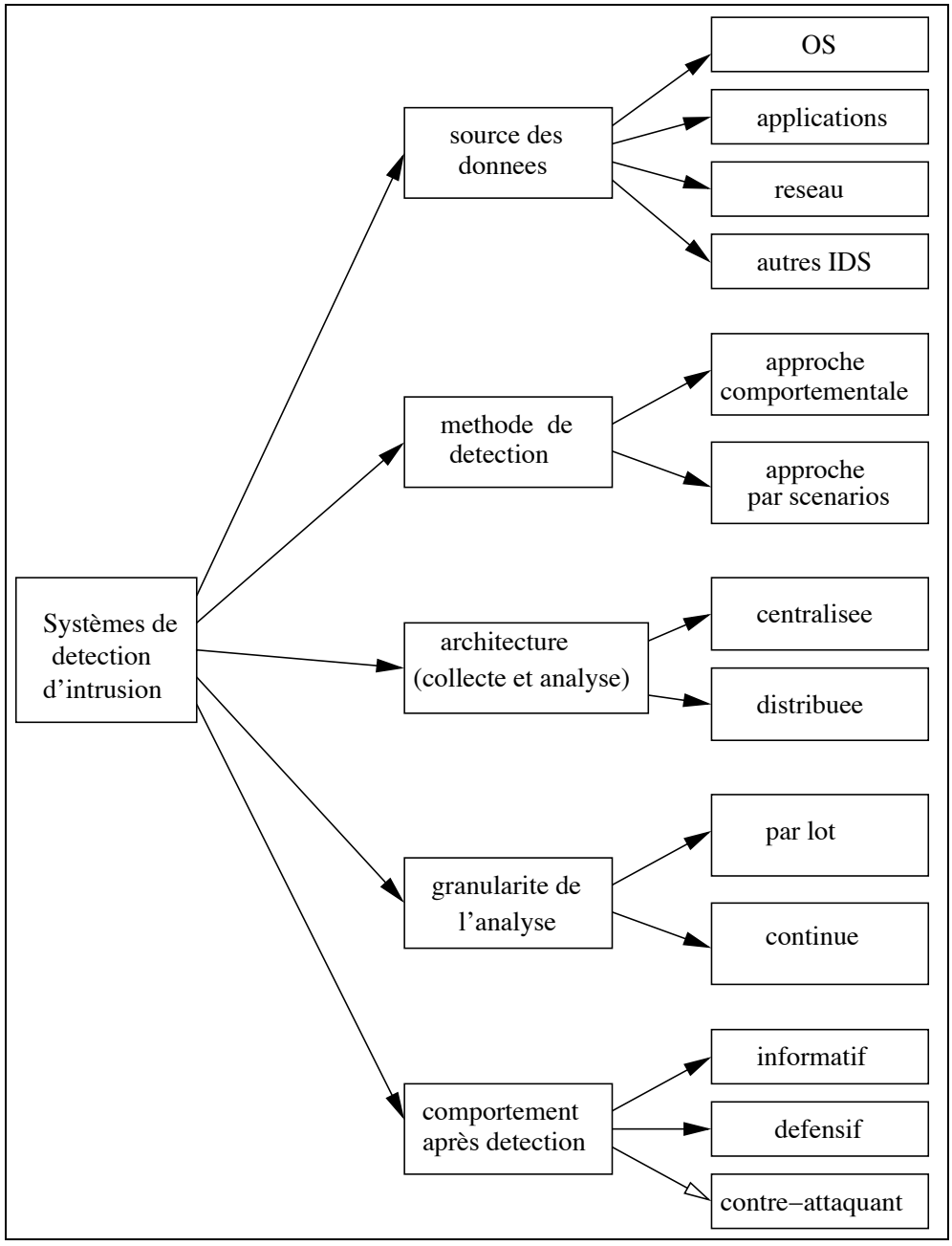


FIG. 2.1 – Une classification des systèmes de détection d'intrusions

(temps CPU, mémoire, espace disque, etc.). Pour autant, une seule source nous paraît réellement fiable : l'audit de sécurité. L'audit permet de définir des événements, de les associer à des utilisateurs et d'assurer leur collecte dans un fichier s'ils surviennent. On peut donc ainsi potentiellement disposer d'informations sur tout se qui se passe sur le système. L'exemple d'un tel système est l'audit BSM, de Sun [Sun00].

Les grandes catégories d'applications savent toutes générer des informations sur la manière dont on les utilise. C'est le cas des serveurs web, par exemple, qui génèrent des fichiers journaux (*logs*) contenant un historique des requêtes qui leur parviennent.

A partir de [HDL+90] on voit apparaître une autre source d'information : le réseau. De nos jours, la plupart des IDS commerciaux utilisent cette source (par le biais de *sniffers*). On parle alors de NIDS, pour *Network-based IDS*. Quelques projets utilisent en outre des informations issues des MIB SNMP[1] (approche proposée initialement par [DGKS94]) car elles présentent l'avantage d'être plus synthétiques.

Enfin, il est intéressant qu'un IDS puisse modifier son comportement si un autre IDS, opérant sur le même système d'information, l'informe d'une attaque en cours. En d'autres termes, les IDS doivent coopérer [MMM+01]. Dans ce contexte, l'IETF normalise actuellement un format d'alerte, baptisé IDMEF (*Intrusion Detection Message Exchange Format*) [CD03].

2.1.2 Avantages et inconvénients des différentes sources

Les données générées par les systèmes d'exploitation, et tout particulièrement par les systèmes d'audit, sont très pertinentes. En effet, une attaque se traduit *in fine* par une action ou une suite d'actions sur le système. C'est le système en lui même qui est le plus apte à générer de l'information sur ces actions.

Pour autant, cette source de données présente un inconvénient qui est parfois rédhibitoire : elle induit une surcharge sur le système surveillé, qui devient parfois importante (quelques dizaines de pour-cent).

Certaines attaques (*port scanning*, déni de service par inondation de requêtes, etc.) sont difficiles à détecter en utilisant une source interne au système d'exploitation. Dans certains cas, ces attaques ne laissent même pas de trace. Ainsi, un déni de service, s'il a réussi, peut avoir fait « tomber » la machine avant que le système d'audit n'ait enregistré les traces de l'attaque dans ses fichiers de log.

Pour certaines attaques, dont le vecteur est le réseau, la source réseau est donc la plus pertinente. En outre, en utilisant un *sniffer* installé sur une machine spécifique, on ne charge pas les machines « utiles ».

Cependant, la source réseau présente aussi des inconvénients : sur les réseaux switchés, la collecte d'information n'est pas aisée ; les débits de plus en plus élevés

[1] *Simple Network Management Protocol Information Management Base*

rendent difficile la capture de l'intégralité du flux réseau ; enfin, si l'usage du chiffrement se généralise, l'analyse des données capturées deviendra impossible, sauf si l'on dispose des clés de déchiffrement, ce qui reste peu probable.

Les deux sources précédentes présentent l'inconvénient d'être sémantiquement pauvres. Dans certains cas, il sera préférable de disposer d'informations de plus haut niveaux. Ainsi, l'invocation d'une fonctionnalité donnée d'un logiciel, accompagnée des paramètres qui lui sont passés, est parfois préférée à la suite des appels système générés par cette invocation. Dans ce cas, on utilisera une source applicative ou des alertes provenant d'un autre IDS, qui aura su agréger l'ensemble des appels système pour détecter que l'invocation de fonctionnalité a eu lieu.

2.2 Principes de détection

Deux approches sont utilisées pour l'analyse des données : l'approche comportementale (*anomaly detection*) [And80] et l'approche par scénario (*misuse detection*) [Sma88] :

- l'approche comportementale consiste à analyser les données dans le but d'y détecter toute déviation par rapport à un comportement de référence, préalablement défini ;
- l'approche par scénario analyse des données d'audit à la recherche de scénarios d'attaques dont les signatures sont prédéfinies et stockées dans une *base de signatures*.

Nous donnons dans la suite de ce paragraphe quelques précisions sur chacune de ces approches[2].

2.2.1 L'approche comportementale

Le comportement de référence peut être construit par apprentissage ou fixé *a priori*. En outre, cette référence peut prendre en compte le temps implicitement ou explicitement.

Prise en compte implicite du temps

— *Approche statistique*

[Den87] propose une approche statistique permettant d'obtenir un modèle de comportement appelé profil. L'outil phare des année 87-93, (N)IDES [JVL+93], s'appuie sur cette approche. Le profil résulte d'une phase initiale d'apprentissage. Il est construit à partir de variables considérées comme aléatoires et échantillonnées à

[2] L'utilisation périodique, par exemple journalière, d'un outil de contrôle d'intégrité tel que *tripwire* peut être vu comme une forme de détection d'intrusions. En effet, l'éventuelle modification de fichier qui peut ainsi être détectée peut être la conséquence d'une intrusion, que l'on détecte donc par là même. Cependant, nous considérons que cette approche sort du cadre de ce mémoire.

intervalles réguliers (c'est en quoi la prise en compte du temps est ici implicite). Ces variables peuvent être le temps processeur utilisé, la durée et l'heure des connexions, etc. Appelons les $X_1, X_2, ..., X_n$.

A chaque variable X_i est associée une densité de probabilité Q_i. L'intervalle des valeurs possibles de X_i est divisé en seize sous-intervalles croissants géométriquement avec un facteur 2 (Q_i est donc constituée de seize fréquences). Ce découpage peut être affiné lorsqu'on acquiert plus d'expérience sur le système cible. Les densités de probabilité sont mises à jour quotidiennement selon un algorithme qui permet de tenir compte du passé en favorisant ce qui est proche.

Les diverses variables aléatoires X_i s'expriment dans des unités quelconques (quantum de temps, nombre d'accès fichier par minute, nombre de pages imprimées par jour, ...). De manière à pouvoir mesurer l'écart entre le comportement observé et le comportement de référence, on doit associer à chaque variable X_i une grandeur D_i qui doit s'exprimer dans une unité commune. On cherche à ce que D_i soit, d'une part proche de zéro lorsque la valeur observée de X_i est normale (c-à-d conforme au passé), d'autre part croissante lorsque la valeur observée de X_i s'éloigne de ce qui est normal. Une solution possible consiste à tout ramener à un nombre d'écarts type relatif à une loi normale réduite. En effet, cette loi constitue une première approximation de toute autre loi de probabilité. On peut pratiquer de la manière suivante :

1. La distribution de probabilité de chaque variable X_i est connue. Supposons par exemple que X_i représente le nombre de mails envoyés par jour par un utilisateur donné et que l'on constate que :
   - 1% des X_i est dans l'intervalle $I_1 = [0, 1[$
   - 30% des X_i sont dans l'intervalle $I_2 = [1, 2[$
   - 42% des X_i sont dans l'intervalle $I_3 = [2, 4[$
   - 24% des X_i sont dans l'intervalle $I_4 = [4, 8[$
   - 3% des X_i sont dans l'intervalle $I_5 = [8, 15]$
2. Par ordre de probabilité croissante, pour N variant de 1 au nombre total d'intervalles, on calcul la somme des N probabilités correspondantes. L'intérêt de ce calcul est patent au point (3). Reprenons l'exemple ci-dessus : $P_1 = 1\%$; $P_2 = 3 + 1 = 4\%$; $P_3 = 4 + 24 = 28\%$; $P_4 = 28 + 30 = 58\%$; $P_5 = 58 + 42 = 100\%$.
3. On calcule alors, pour une loi normale réduite, (c-à-d une gaussienne de moyenne nulle et d'écart type unité) les valeurs D_i telles que la probabilité qu'une variable Y suivant cette gaussienne ne soit pas dans l'intervalle de confiance $[moyenne \pm D_i \times ecarttype]$ soit égale à P_i. Soit :

   $$1 - Prob.(m - D_i \times e < Y < m + D_i \times e) = Pi$$

   En reprenant l'exemple ci-dessus, on obtient : $D_1 = 2.58$; $D_2 = 2.06$; $D_3 = 1.9$; $D_4 = 0.56$; $D_5 = 0$.

A l'issue de la phase d'apprentissage, on passe en phase de détection. Lors d'une nouvelle observation, la valeur D_j est associée à X_{n+1} si X_{n+1} tombe dans le j$^{\text{ème}}$ intervalle dans l'ordre croissant de probabilité (par exemple, si X_{n+1} est dans l'intervalle [0, 1[, on lui associe 2,58 ; si X_{n+1} est dans l'intervalle [1, 2[, on lui associe 0,56 ; etc.). On peut alors calculer de la manière suivante l'écart entre le comportement appris et ce nouveau comportement observé :

$$EC = (D_1 D_2 ... D_n) C^{-1} (D_1 D_2 ... D_n)^t$$

où $(D_1 D_2 ... D_n)^t$ est la transposée de la matrice ligne $(D_1 D_2 ... D_n)$ et où C^{-1} est l'inverse de la matrice de covariance de $(D_1 D_2 ... D_n)$. Cette matrice se justifie car les variables X_i sont dépendantes les unes des autres.

Ainsi, plus la nouvelle observation est proche du comportement de référence, plus l'écart de comportement est faible (il est même nul dans le cas où tous les X_i tombent dans l'intervalle pour lequel leur probabilité est la plus forte).

— *Approche à base de règles*

[VL89] utilise l'historique des commandes et des accès fichier pour construire des règles d'usage liées à chaque utilisateur.

Prise en compte explicite du temps

La phase d'apprentissage peut être utilisée, non plus pour construire un profil, mais pour déterminer des séquences d'actions qui sont courantes pour l'entité à modéliser. C'est ce que propose par exemple [DBS92], qui utilise des réseaux de neurones, ou [FHS97], pour laquelle la phase d'apprentissage consiste à observer un service pendant un certain temps afin de construire une base des séquences d'appels système normales. Par analogie avec le système de défense immunitaire biologique, basé sur le principe de reconnaissance de cellules étrangères, l'approche proposée consiste à comparer, en phase de détection, le comportement observé au comportement de référence : toute séquence étrangère est considérée comme une potentielle exploitation d'une faille de sécurité du service.

A titre d'exemple, considérons que la suite des appels système observés en phase d'apprentissage soit la suivante : $ABCDBCDBABC$. On choisit une largeur N de fenêtre glissante et on constitue, à partir de la séquence observée, une base de N-grams. Si N vaut par exemple 4, on obtient la base de patterns suivante : $ABCD$, $BCDB$, $CDBC$, $DBCD$, $BCDB$, etc. En phase de détection, les séquences de taille N sont comparées à celles de la base de comportements. Si la séquence observée n'est pas dans la base, il y a anomalie dans le comportement. Il n'y a pas pour autant forcément émission d'une alerte : le nombre de séquences anormales observées et le temps écoulé entre ces anomalies déterminent le déclenchement d'une alerte.

Construction d'une référence a priori

Il a également été proposé de ne pas construire le comportement de référence par apprentissage, mais de le fixer *a priori*.

La référence [Sma88] propose d'affecter à chaque utilisateur un profil correspondant *a priori* à sa catégorie (enseignant ou étudiant, en prenant l'exemple d'un environnement académique). Le profil est dans ce cas similaire à ce que l'on a décrit ci-dessus, mais ne résulte pas d'un apprentissage.

Dans [RTG94, Pax98], le comportement de référence est fixé par un ensemble d'actions autorisées ou interdites (on parle parfois d'approche orientée politique, *policy based intrusion detection*). Dans ces deux références, est en fait proposée une approche de type *firewall* : un flux de paquets réseau est observé et tout événement en contradiction avec les règles de la politique est signalé comme intrusif.

Bien évidemment, la même approche peut être retenue pour un flux d'événements de type quelconque. Ainsi, sur un système d'exploitation, la politique de sécurité est implémentée, dans le cas le plus courant grâce à un service d'authentification et à un service de contrôle d'accès. Les droits d'accès constituent la base nécessaire à la mise en œuvre au niveau système de l'approche orientée politique. Paradoxalement, alors que la définition accepté par tous du terme « intrusion » est « toute violation de politique de sécurité », une telle approche n'a pas à notre connaissance été explorée. Cela a motivé notre travail sur les flux de références [ZMB02, ZMB03a, ZMB03b], que nous présentons dans le paragraphe 3.3.

2.2.2 L'approche par scénarios

L'approche par scénarios consiste à rechercher, dans les données à analyser, des signatures d'attaques que l'on connaît.

Quatre grandes approches ont été proposées à cet effet : l'approche par recherche de patterns (proposée initialement par [HDL+90]), l'approche à base de règles (proposée initialement par [LJ88]), l'approche par séquences temporelles (proposée initialement par [Por92, Ilg93]), l'approche par langage de corrélation d'événements ([LWJ98] est la première référence à proposer un tel langage).

Nous présentons successivement ces 4 approches dans la suite de ce paragraphe.

— *Approche par recherche de patterns*

Le terme franglais « recherche de patterns » est ici préféré à « analyse de séquences », car cette dernière appellation laisserait supposer qu'on parle de séquences d'événements à analyser. En fait, ce n'est pas le cas. Ce qui est recherché, c'est une séquence d'informations particulières (que nous appelons donc ici *pattern*) au sein d'un *unique* événement d'audit.

En d'autres termes, la signature d'une attaque correspond à un pattern particulier qu'il faut rechercher dans chaque événement. On a donc là une technique semblable à celle utilisée par les anti-virus. Il en découle un inconvénient critique : une telle signature, généralement trop simpliste, peut être trouvée dans des événements qui ne sont pas révélateurs d'attaque. On a donc un gros risque de faux positifs. En revanche, cette technique présente l'avantage d'être très efficace en terme de temps de traitement si l'algorithme de pattern matching utilisé est correct (souvent, un algorithme de type Moyer-Moore [BM77] est utilisé).

— *Approche à base de règles*

L'idée consiste à coder les attaques sous la forme de règles condition-action. Les conditions portent sur l'état du système surveillé et la nature des événements analysés ; les actions permettent, soit de mémoriser le nouvel état du système, soit de conclure à la présence d'une attaque. Bien évidemment, des systèmes experts sont utilisés pour implanter cette approche.

Cette approche est intéressante car elle permet de tirer partie de la puissance des systèmes experts. En particulier, la capacité d'unification des variables est ici particulièrement utile pour détecter diverses variantes de scénarios.

Bien que séduisante, cette approche pose en pratique un problème d'efficacité si le débit du flux d'événements à analyser est trop important, ou si le nombre de règles activables à un moment donné est trop important. Cette dernière difficulté rend l'écriture des règles délicate.

— *Approche par séquences temporelles*

L'idée consiste à coder les attaques sous la forme d'une suite d'états par lesquels passe le système à surveiller, depuis un état initial sûr, jusqu'à un état final compromis [Por92, Ilg93]. Les états sont définis par des conditions sur les variables du système (ex : exists(object) = true ; owner(object) <> root), les transitions par des actions dont les événements à analyser sont l'image (ex : chmod(object)). L'analyse permet donc de passer d'un état à un autre en fonction des événements qui surviennent ; une alerte est émise si l'état final est atteint.

[KS94] généralise cette idée en proposant l'utilisation de réseau de pétri. Le principal apport de cette utilisation réside dans la capacité à exprimer des signatures plus complexes. Par exemple, il sera possible d'exprimer que deux sous-séquences peuvent survenir dans un ordre quelconque (les événements de ces deux sous-séquences pouvant être entremêlés), mais que la survenue de l'une **et** de l'autre est nécessaire pour passer dans un état donné.

On le voit, l'approche est très proche de celle à base de règles[3]. Les avantages et inconvénients sont donc les mêmes.

— *Approche orientée langage de corrélation d'événements*

Nous l'avons souligné en présentant l'approche par recherche de pattern : si la signature d'une attaque est trop simpliste, elle peut conduire à de nombreux faux positifs. Pour contourner ce problème, il faut être capable de définir, puis de rechercher dans le flux d'événements à analyser, des suites d'événements corrélés logiquement et temporellement. C'est ce que propose de faire l'approche à base de règles et celle par séquences temporelles. Cependant, ces deux dernières approches reposent sur des algorithmes exploitant des systèmes de transitions ou de déduction en chaînage avant (c'est aussi le cas des langages de ASAX [Mou97] ou de P-BEST [LP99]). Les signatures décrites expriment comment reconnaître une suite d'événements et spécifient une action à effectuer lorsqu'un événement intéressant est rencontré. De telles signatures sont donc très dépendantes de l'algorithme de reconnaissance sous-jacent, ce qui en limite la réutilisabilité.

D'autres langages ont été proposés[4]. Ce sont des langages dits « de détection ». On préférera les appeler « langage de corrélation d'événements ». Ils permettent d'exprimer, indépendamment de l'algorithme de détection, des contraintes sur les événements dont l'occurrence est révélatrice d'une intrusion **et** des contraintes entre ces événements. Trois types de contraintes doivent pouvoir être exprimées :

- Contraintes de sélection (filtrage) : elles permettent d'identifier les événements intéressants dans un flux d'événements à analyser. Ces événements sont ceux qui correspondent à une étape de l'intrusion que l'on souhaite détecter. Ces

[3] D'ailleurs, [Por92] montre une mise en œuvre utilisant un système expert.

[4] De nombreux langages existent aujourd'hui autour de la détection d'intrusions. Pour clarifier les choses, [VEK00] classe ces langages en 6 catégories :

- les langages d'*exploit* servent à décrire ce qu'il faut faire pour réaliser l'attaque ; c'est la vision de l'attaquant [Sec98, Der99, LMPT98] ;
- les langages d'événements servent à décrire le format des événements à analyser [Bis95] ;
- les langages de détection servent à décrire les symptômes de l'attaque [LWJ98, EVK00, CO00, LTL01, MM01, PD01].
  Remarque 1 : les chroniques [DGG93, Dou94, Dou96], conçues initialement dans le monde de la surveillance de réseau, ont été utilisées en détection d'intrusions (pas de publication) ; elles nous paraissent très prometteuses par la richesse de leur expressivité.
  Remarque 2 : il faut aussi noter que certains de ces langages peuvent être également vus comme de langages de corrélation d'alertes (voir ci-dessous). C'est notamment le cas de Lambda [CO00] et ADeLe [MM01] ;
- les langages de description d'alertes servent à décrire l'alerte qui doit être émise si l'attaque est détectée [FKP+99, CD03] ;
- les langages de corrélation d'alertes servent à décrire les liens qui peuvent exister entre différentes alertes et qui seraient donc révélateurs d'une vague d'attaques en cours [TL00, CO00, MM01] ;
- les langages de réaction servent à décrire les éventuelles contre-mesures à prendre après détection [MM01].

contraintes portent sur la valeur d'un champ ou les valeurs de plusieurs champs d'un unique événement.
- Contraintes de corrélation logique : elles définissent des contraintes sur les valeurs portées par différents événements. La forme la plus simple de corrélation logique consiste à fusionner en un seul événement, N événements identiques (action de comptage).
- Contraintes d'ordonnancement (corrélation temporelle) : elles définissent un ordre total ou partiel que doivent respecter les événements retenus par le filtrage.

Bien évidemment, quelque soit le langage, les signatures exprimées sont ensuite exploitées par un algorithme de détection. L'indépendance entre la signature et l'algorithme qui l'exploite a une limite : l'efficacité en temps (rapidité de détection) et en espace (nombre d'hypothèses à traiter en parallèle) de l'algorithme de détection[5]. Ce soucis d'efficacité impose, dans un nombre de cas qui doit rester le plus faible possible, de déroger au principe d'indépendance entre signature et algorithme ; on sera ainsi amené à offrir dans les langages des constructions permettant au concepteur d'une signature de limiter l'explosion combinatoire du nombre d'hypothèses à conduire en parallèle.

Pour conclure, si des langages commencent à voir le jour, l'approche en elle même reste à valider : est-il réaliste de vouloir corréler des événements d'audit lorsque le flux d'événements et le nombre de signatures sont importants ? La recherche d'une réponse à cette question motive notre travail autour du langage ADeLe [MM01]. On sera peut-être amené à limiter les formes de corrélations appliquées aux événements aux formes les plus simples (filtrage et comptage), tandis qu'on appliquera les corrélations plus complexes (logique et temporelle) aux alertes.

2.2.3 Avantages et inconvénients des deux approches

Le tableau 2.1 résume les avantages et inconvénients des deux approches.

L'approche comportementale

Lorsqu'elle s'appuie sur un mécanisme d'apprentissage, l'approche comportementale suppose qu'une intrusion implique un usage inhabituel du système surveillé. C'est sans doute vrai dans de très nombreux cas, mais il reste sûrement des cas où ce n'est pas vrai.

Un outil basé sur une telle approche génère une alerte dès qu'il détecte un comportement trop éloigné des comportements appris. Or, la déviation du comportement observé peut être due à une évolution naturelle de l'environnement et du système, ou au fait que le comportement de référence n'est pas exhaustif : c'est un faux positif. De plus, si des attaques ont été commises durant la phase d'apprentissage, elles seront considérées comme normales : c'est un faux négatif. Un faux

[5]Nous avons déjà noté que si la plupart des outils utilisés aujourd'hui ne propose que de la recherche de patterns au sein d'un unique événement, c'est pour des raisons d'efficacité.

négatif peut également se produire si l'attaquant (utilisateur interne malicieux) modifie lentement son comportement afin de parvenir à un comportement intrusif qui, ayant été progressivement appris, ne sera pas détecté.

Plusieurs études ont en outre montré les limites d'un modèle de comportement d'utilisateur. [Deb93] montre que dès que ce dernier n'est pas un novice, son comportement est trop riche pour être modélisé statistiquement, quasiment tout étant possible à tout moment. En d'autres termes, pour la plupart des utilisateurs, le modèle statistique conduit à reconnaître toutes les actions comme normales. C'est pourquoi on s'intéresse plutôt aujourd'hui à la modélisation d'entités au comportement plus simple et reproductible que celui d'un utilisateur.

Lorsqu'elle ne s'appuie pas sur un mécanisme d'apprentissage (approche orientée politique), l'approche comportementale ne présente pas ces inconvénients.

Pour autant, avec ou sans apprentissage, l'approche comportementale pose un problème de caractérisation des alertes émises. En effet, les alertes sont émises en cas de déviation de comportement ou de violation de politique. En aucun cas n'est donnée la cause de cette déviation ou violation. En d'autres termes, un IDS implantant un telle approche laisse entièrement à la charge de l'administrateur de sécurité la tâche de diagnostic.

On le voit, l'approche comportementale n'est donc pas exempte de défauts intrinsèques. En revanche, elle présente un avantage très important dans la pratique, qui justifie que l'on s'y intéresse de près : elle ne nécessite pas de base de signatures d'attaque, très difficile à construire et, surtout, à maintenir. Corollairement, elle permet de détecter des attaques inconnues, pour peu que ces attaques impliquent un fonctionnement inhabituel (ce qui est souvent le cas) ou une violation de la politique.

L'approche par scénarios

L'approche par scénarios présente un inconvénient majeur : on ne peut détecter que des attaques connues et référencées dans la base. Il faut mettre à jour cette base au même rythme que celui de l'apparition de nouvelles attaques, c'est-à-dire quasiment quotidiennement.

Cette approche, par le biais de la base de signature, prend en compte les comportements exacts des attaquants potentiels. En théorie, un détecteur par scénario devrait donc présenter un taux de faux négatif très faible (théoriquement nul). Malheureusement, pour des raison de performances, les outils actuels se limitent à de la recherche de pattern au sein d'un unique événement à analyser ; c'est une approche simpliste qui conduit à la génération de nombreux faux positifs. Le seul avantage de l'approche n'est donc pour le moment que théorique.

	Avantages	Inconvénients
Comportementale	- Pas besoin d'une base d'attaque - Détection d'intrusions inconnues possible ⇒ **peu de faux négatifs**	- Pour un utilisateur au comportement riche, toute activité est normale ⇒ **risque de faux négatifs** - En cas de profonde modification de l'environnement du système cible, déclenchement d'un flot ininterrompu d'alertes ⇒ **risque de faux positifs** - Utilisateur pouvant changer lentement de comportement dans le but d'habituer le système à un comportement intrusif ⇒ **risque de faux négatifs**
Par scénarios	- Base de signature ⇒ **peu de faux positifs** si les signatures sont suffisamment précises	- Base de scénarios difficile à construire et, surtout, à maintenir ⇒ **risque de faux négatifs** - Pas de détection d'attaques non connues ⇒ **risque de faux négatifs**

TAB. 2.1 – Avantages **théoriques** et inconvénients de l'approche comportementale et de l'approche par scénarios

2.3 Architecture

Selon la définition donnée dans le chapitre d'introduction, un IDS est constitué d'un ou plusieurs capteurs, un ou plusieurs analyseurs et un ou plusieurs managers. Le nombre et l'emplacement de ces composants définissent l'architecture de l'IDS. Selon les architectures, la collecte, d'une part, l'analyse des données, d'autre part, peuvent être centralisées ou distribuées.

L'architecture la plus classique, adopté par la quasi-totalité des outils aujourd'hui, est constituée de plusieurs sondes (sur les différents brins du réseau et les différentes machines du système d'information) qui collectent et analysent leurs propres données et émettent des alertes vers un unique manager. Généralement, capteur et analyseur résident sur la même machine et il y a un seul capteur et un seul analyseur. Le manager se contente d'une gestion simple des alertes reçues (classement par degré de gravité supposé, par exemple). Dans ce cas, on a en fait une juxtaposition de collectes et analyses centralisées. On ne peut selon nous pas parler d'IDS distribué, contrairement à ce qui est généralement avancé dans la littérature. Nous préférons parler d'approche hiérarchique.

En conservant la même architecture, on peut selon nous parler d'approche distribuée si l'on exploite le caractère récursif du modèle IDWG. En effet, un manager, au lieu de se contenter de les classer, peut mettre en place des techniques de corrélation d'alertes. Le manager peut ainsi, par exemple, détecter une attaque complexe impliquant plusieurs étapes qui auront quant à elles été détectées par les sondes. Une telle architecture est pour le moment peu répandue. Elle a cependant été utilisée dans le projet MIRADOR [Equ00], auquel nous avons participé.

Enfin, de nombreuses références proposent d'utiliser des agents mobiles pour la détection d'intrusions (voir par exemple [JMaM99] ou [KT02]). De notre point de vue, ce n'est là qu'une manière, certes fort intéressante, de mettre en place une approche distribuée. A la réception d'une sollicitation provenant d'une sonde, le manager envoie un agent sur la machine sur laquelle tourne la sonde (une plate-forme agent adéquate doit donc résider sur cette machine). L'agent peut alors assurer une analyse fine des données de la sonde. L'avantage de cette approche provient du fait qu'on déplace de petits bouts de code (les agents), au lieu de déplacer des quantités de données plus importantes. C'est une technologie que nous avons utilisée dans [ACP+02].

2.4 Granularité de l'analyse

Deux modes de fonctionnement sont envisageables : une analyse continue des données fournies par les capteurs ou une analyse par lot de ces mêmes données.

Ces deux modes de fonctionnement sont souvent confondus avec la capacité des IDS à traiter, ou non, les données en temps réel. Il est vrai que les systèmes qui

travaillent par lot (ce sont souvent des prototypes de recherche) ne visent généralement pas le temps réel. *A contrario*, les systèmes commerciaux, qui cherchent à fonctionner en temps réel, analysent les données en continu (mais, on l'a vu, en utilisant des techniques simples de recherche de patterns). On peut pourtant noter qu'un système peut traiter les données en continu et fournir son diagnostic avec un délai très important, tandis qu'un autre système traitera les données par lot de petite taille et fournira son diagnostic beaucoup plus rapidement, idéalement pendant le déroulement de l'attaque[6], pour pouvoir la stopper ou en limiter les conséquences.

Dans le traitement par lot, l'idée est d'accumuler de l'information, pour en faire périodiquement une analyse globale (toutes les minutes, heures, jours, semaines et mois par exemple). Bien sûr, chaque analyse devra se faire le plus efficacement possible, en temps et en espace.

Afin d'éviter de rater certaines attaques qui chevaucheraient plusieurs fenêtres, soit on rend la fenêtre d'analyse glissante, soit on analyse les données en continu : dans le premier cas, rien ne garantit que le pas retenu pour le décalage de la fenêtre permette de ne manquer aucune attaque ; dans le deuxième cas, on doit « oublier » ce qui s'est passé avant une certaine date/heure, pour ne pas se heurter à un problème d'efficacité en espace, et l'on retombe donc sur un problème si l'attaque s'étale sur une période plus longue.

On le voit, traitement par lot et traitement continu sont en fait très semblables. Ils sont décorrélés de la capacité de traitement en temps réel et ils présentent des problèmes identiques.

2.5 Comportements en cas de détection

Une dernière caractéristique importante des systèmes de détection d'intrusions concerne leur capacité de réaction en cas de détection d'une attaque. Trois réactions sont envisageables : informer l'administrateur (approche dite passive), tenter de stopper l'attaque ou contre-attaquer (approche dite active).

L'approche passive consiste à émettre une alerte en direction du manager. C'est ce dont se contentent généralement les prototypes de recherche, qui ne visent « qu'à » valider des méthodes de détection. Certains outils commerciaux permettent aussi d'émettre un courrier électronique, voire un mini-message de type SMS, vers l'administrateur système.

La plupart des IDS commerciaux offrent en outre la possibilité d'apporter une réponse active à l'intrusion. Il s'agit de prendre automatiquement des mesures pour stopper l'attaque en cours (coupure des connexions réseau, reconfiguration dyna-

[6] C'est d'ailleurs la définition que l'on peut donner de « temps réel » dans le domaine de la détection d'intrusions.

mique du firewall pour bloquer le trafic issu de la source présumée de l'attaque, suppression des processus système en cause, etc.) ou pour empêcher son renouvellement (reconfiguration du firewall, suppression de la vulnérabilité exploitée par l'attaque, retour à un état antérieur à l'attaque, etc.). Il faut noter que l'importance actuelle de la proportion de faux positifs rend dangereuse l'approche active. Ainsi, rien ne garantit que l'adresse dont semble provenir l'attaque est bien celle que l'on croit.

Enfin, pour la même raison (importance des faux positifs), répondre par une contre-attaque nous parait difficilement envisageable.

2.6 Positionnement de nos travaux

Le tableau 2.2 positionne nos différents travaux et publications, en fonction des grandes caractéristiques des IDS présentés dans les sections précédentes. Cette section donne quelques détails, caractéristique par caractéristique.

2.6.1 Vis-à-vis des sources de données

Nous avons principalement utilisé des données en provenance du système d'exploitation. Dans [Mé93, Mé98, KNMH00], nous utilisons le système d'audit [Mé93, Mé98, KNMH00]. Il nous semble que ce sont là des données pertinentes, car une attaque se traduit *in fine* par une action ou une suite d'actions sur le système et que ce sont ces actions qu'il faut analyser. Dans [ZMB02, ZMB03a, ZMB03b], l'IDS réside au cœur du système d'exploitation en traitant chaque appel système (voir paragraphe 3.3 de ce mémoire).

Dans le cas de [MMVM00], il s'agit de modéliser le comportement d'objets CORBA. Dans ce cas, les requêtes échangées entre objets sont les données les plus pertinentes, car sémantiquement les plus riches. En modélisant la suite de requêtes et la répartition de leurs paramètres, nous obtenons une représentation fidèle du comportement au niveau applicatif. Dans ce cas, les données ont été obtenues par le biais d'intercepteurs CORBA.

Le taux de faux positifs constitue aujourd'hui le problème majeur des IDS. Le réduire est essentiel. Pour cela, la coopération entre IDS et la corrélation d'alerte est sans doute déterminant. A ce titre, il faut être capable de traiter des alertes en provenance de divers IDS. C'est un des objectifs du modèle de données proposé dans [MMDD02].

2.6.2 Vis-à-vis de la méthode de détection

Nous avons principalement œuvré dans le domaine de l'approche par scénarios. Pour autant, l'approche comportementale nous semble également prometteuse, voire incontournable dans certains cas et nous avons souhaité nous y investir également.

Source de données	**Travaux publiés**
Système d'exploitation	[Mé93, Mé98, KNMH00, ZMB02, ZMB03a, ZMB03b]
Application/service	[MMVM00]
Réseau (SNMP v3)	[ACP+02]
Alerte d'IDS	[MMDD02]

Type de détection	**Mécanisme**	**Travaux publiés**
Comportementale	Temps implicite	[PMM02]
	Temps explicite	[MMVM00]
Par scénarios	Recherche de pattern	[KNMH00]
	Règles	
	Séquences temporelles	[Mé93, Mé94, Mé98, KNMH00]
	Langage de corrélation	[MM01, MMH01]

Architecture	**Travaux publiés ou réalisés**
Centralisée	Mono-capteur/mono-analyseur [Mé93, Mé94, ZMB02, ZMB03a, ZMB03b]
	Multi-capteurs/mono-analyseur [Mé98, MMVM00]
Distribuée	[Equ00]
Agents mobiles	[ACP+02, PPMC03, PPM+03]

Analyse	**Travaux publiés ou réalisés**
Par lot	[Mé93, Mé94, Mé98, KNMH00, MMVM00]
Continue	[Equ00], [ZMB02, ZMB03a, ZMB03b]

Réaction	**Travaux publiés ou réalisés**
Informatif	Tous nos travaux
Défensif	Un élément de [MM01]
Contre-attaquant	

TAB. 2.2 – Positionnement de nos travaux, relativement aux sources de données, à la méthode de détection, à l'architecture de l'IDS, à la granularité de l'analyse et au comportement après détection. La même référence peut apparaître pour plusieurs caractéristiques.

Approche par scénarios

Dans les travaux que nous avons initiés en thèse puis poursuivis par le développement de l'IDS G$^{A}_{S}$SA$_{T}$A [Mé93, Mé94, Mé98], nous avons utilisé des algorithmes génétiques [Gol89], dans le but de rechercher la présence, dans des fichiers d'audit système, d'ensembles prédéfinis d'événements, chaque ensemble constituant une signature d'attaque.

Ces travaux ont été conduits dans le but de répondre au problème suivant : comment, alors que le nombre de signatures d'attaque peut être important et que les données d'audit sont très volumineuses, déterminer le plus rapidement possible[7] la présence d'une ou plusieurs signatures dans les données d'audit ?

Pour répondre à cette question, les événements des signatures ne sont pas ordonnés. Les signatures sont des ensembles de couple (événement, nombre d'occurrences de l'événement). On ne tient donc pas compte de l'ordre d'apparition des événements dans les données d'audit. La perte de l'ordre temporel altère la pertinence du diagnostic (risque de faux positifs) mais peut constituer un avantage dans la pratique si l'ordonnancement total des événements est impossible.

En tout état de cause, l'approche proposée doit être considérée comme un filtrage initial : on ne retient des données d'audit que celles dans lesquelles une signature d'attaque est potentiellement présente. Reste à déterminer, par exemple par une approche plus classique de pattern matching, si l'ordre temporel des événements constitutifs de la signature est respecté. Le filtrage initial permet d'envisager une application efficace des algorithmes de pattern matching.

Les travaux présentés dans [KNMH00] se placent dans la même lignée : être capable de déterminer le plus rapidement possible la présence d'attaques potentielles. Le filtrage initial est cette fois réalisé par un algorithme spécialement conçu d'appariements multiples avec approximation. L'ordre temporel est donc cette fois pris en compte dès le filtrage.

Dans cet appariement approximatif, la seule opération à envisager est l'insertion (les événements constitutifs de la signature ne sont pas forcément contigus dans le fichier d'audit). Appelons k le nombre d'insertions autorisées. Nous montrons qu'il est possible de déterminer une valeur de k pour laquelle la probabilité d'appariement est très proche de 1 (cette valeur de k dépend des signatures considérées). Il est donc possible, au vu d'une base de signatures, de fixer une valeur de k permettant de filtrer les données d'audit sans pour autant risquer de faux négatifs dus à un filtrage trop fort.

Pour valider expérimentalement ce résultat, nous nous sommes intéressés à un fichier d'audit de 25000 événements en considérant 3 signatures de 4, 10 et 4 événements, présentes respectivement à raison de 1, 2 et 4 occurrences. Les résultats montrent qu'avec la valeur de k obtenue par l'étude, il est possible de filtrer 85% du fichier d'audit sans risque de faux négatif. En d'autres termes, seuls 15% de ce fichier sont à analyser par un algorithme plus fin.

[7] Encore une fois, si les IDS utilisés aujourd'hui ne proposent que de la recherche de patterns au sein d'un unique événement, c'est pour des raisons d'efficacité. Ce problème reste donc toujours d'actualité.

Pour cette expérimentation, nous nous sommes intéressés à des commandes système. Un *mapping* est réalisé *off-line* entre ces commandes et les lettres d'un alphabet sur lequel travaille ensuite l'algorithme d'appariements multiples avec approximation. Nous avons également réalisé une maquette sous linux dans laquelle le mapping est fait dès la génération des événements d'audit. Dans cette maquette, nous nous intéressons aux appels système (utilisation de Snare [sna03]).

Filtrer les événements inintéressants constitue une première étape. Il faut ensuite analyser les événements restants, afin de déterminer s'ils correspondent précisément à une signature de la base. Cette seconde analyse demande une définition précise des signatures. A cet effet, nous avons proposé un langage, ADeLe (*Attack Description Language*) [MM01, MMH01].

Le langage ADeLe a été conçu pour permettre la description la plus complète possible des attaques. Son but premier est de combiner, dans une unique description de haut niveau, toutes les informations disponibles sur une attaque donnée. ADeLe est donc un langage qui ne se restreint pas à la description de signatures, mais permet plus généralement d'exprimer les 6 facettes suivantes : (1) les actions réalisées par l'attaquant pour mettre en œuvre l'attaque (l'exploit), (2) le format des événements qui seront observés suite aux actions de l'attaquant, (3) la signature de l'attaque, (4) les corrélations entre les événements de la signature, (5) l'alerte à générer si l'attaque est détectée, (6) la réponse à apporter en cas d'occurrence de l'attaque décrite.

ADeLe a pour ambition de permettre l'écriture de descriptions possédant les propriétés suivantes :

- Lisibilité : une description ADeLe est constituée de 3 parties. La première décrit l'attaque du point de vue de l'attaquant (*exploit* et événements afférents), la seconde décrit l'attaque du point de vue du défenseur (signature, corrélation et alerte), la dernière partie décrit la réponse à apporter. Une syntaxe proche de XML est utilisée. L'alerte et les événements sont conformes au standard IDMEF[CD03].
- Généricité : la qualité d'une signature est bien évidemment déterminante pour la bonne détection par les IDS de l'attaque correspondante : trop simple, elle conduit à un grand nombre de faux positifs, trop « rigide », elle ne permet pas de prendre en compte toutes les variantes possibles dans l'enchaînement des étapes de l'attaque et peut ainsi conduire à des faux négatifs. Nous avons donc défini un ensemble d'opérateurs de signature qui permet d'écrire des signatures complexes, mais non rigides.
- Modularité : il est possible, au sein d'une description, de faire appel à une autre description. Des paramètres permettent de passer des informations d'une description à l'autre (l'adresse IP cible de l'attaque, pour ne donner qu'un exemple).

Le langage ADeLe est actuellement utilisé pour la configuration de G^{N}G (voir paragraphe 2.6.3), la nouvelle version de G^{A}SSA$_{T}$A, l'IDS développé au sein de l'équipe à la suite de mes travaux de thèse.

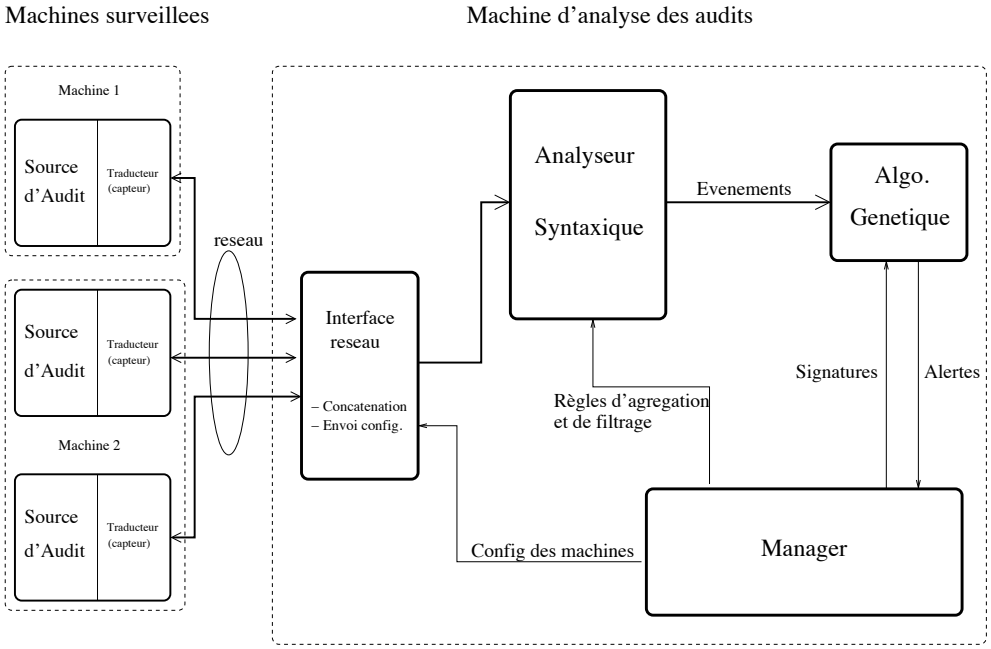


FIG. 2.2 – Architecture de l'outil GASSATA

Approche comportementale

Dans [MMVM00], nous nous intéressons à la détection d'intrusions en environnement objets répartis de type CORBA. Compte tenu des spécificités de cet environnement, une base de signatures se révèle difficile à construire.

Nous proposons donc une approche comportementale dans laquelle sont modélisées, sous forme arborescente, des séquences de taille variable de requêtes CORBA, prises entre deux événements particuliers (connexion et déconnexion).

L'approche proposée est inspirée de [FHS97]. Cependant, les patterns de comportement (suites temporelles des requêtes invoquées) sont de taille variable, ce qui permet une meilleure adéquation aux comportements à modéliser. En outre, l'application d'un modèle de mélange de gaussienne permet de prendre en compte la dispersion et les corrélations entre les différents paramètres des invocations de méthode [PMM02]. L'ensemble constitue donc une approche hybride temps implicite-temps explicite.

2.6.3 Vis-à-vis de l'architecture de l'IDS

Dans certains de nos travaux, l'analyse des données est réalisée de manière centralisée par un unique analyseur résidant sur la même machine que le manager (à titre d'exemple, la figure 2.2 donne l'architecture de GASSATA.). Il nous faut ce-

pendant distinguer deux cas : l'analyseur peut être alimenté par un [Mé93, Mé94, ZMB02, ZMB03a, ZMB03b] ou plusieurs capteurs [Mé98, MMVM00]. Dans ce dernier cas, les capteurs sont répartis sur plusieurs machines.

Dans le cadre du projet MIRADOR [Equ00], nous avons mis en place une approche distribuée, baptisée G^{N}G (G^{A}$_{S}$SA$_{T}$A *New Generation*).

Les signatures d'attaque sont exprimées en ADeLe. L'analyse des données est hiérarchisée en deux niveaux (on voit là clairement apparaître l'aspect récursif du modèle IDWG illustré par la figure 1.1).

Au premier niveau, on analyse chaque événement et on émet une alerte s'il y a appariement avec l'un des événements définis dans les signatures. Ce premier niveau d'analyse est en fait similaire à l'analyseur syntaxique de G^{A}$_{S}$SA$_{T}$A. En revanche, afin de limiter le trafic réseau, il ne réside plus sur la machine d'analyse des données d'audit, mais a été déplacé vers les machines surveillées. Les alertes émises par les analyseurs de premier niveau sont au format IDMEF.

Ces alertes[8] sont ensuite corrélées par un algorithme simple qui remplace l'algorithme génétique. Les contraintes de corrélation sont exprimées dans les signatures ADeLe.

Nous avons défini dans le paragraphe 2.2.2 trois types de contraintes de corrélation : les contraintes de sélection ou de filtrage, les contraintes de corrélation logique, les contraintes d'ordonnancement ou de corrélation temporelle. Dans G^{N}G, les contraintes de filtrage sont traitées par les analyseurs de niveau 1 ; celles de corrélation logique par l'analyseur de niveau 2 (les comptages, simples, devraient être ramenés dans l'analyse de niveau 1) ; les contraintes temporelles ne sont pas traitées.

Dans le cadre du projet RNRT DICO[9], nous sommes cependant en train d'implanter le traitement des contraintes temporelles. A cette occasion, le moteur de détection est entièrement revu ; il fonctionne à présent en une seule passe. Nous n'avons pas encore soumis de publication sur les résultats de ces travaux, qui sont toujours en cours. Un travail important reste en effet à faire : l'évaluation des performances du nouveau moteur dans le cas où la base de signatures est importante.

Les travaux que nous présentons dans [ACP^{+}02, PPMC03, PPM^{+}03] s'intéressent au problème de la détection des intrusions en environnement réseau ad hoc (MANET, *Mobile Ad Hoc NETworks*) ou spontané. Un réseau ad hoc est un réseau sans fil capable de rendre transparentes pour les utilisateurs mobiles les modifications de topologie qu'il subit. Les modifications de topologie, la décentralisation des fonctions réseau, la nécessité de limiter la consommation des équipements et l'usage de la bande passante, sont autant de contraintes spécifiques dont il faut tenir compte pour la conception d'un IDS adapté. L'utilisation d'une plate-forme à agents mobiles répond bien à ces contraintes.

[8] ainsi que d'autres alertes provenant de sondes commerciales ou du domaine publique (Snort)

[9] Détection d'Intrusion COopérative ; voir l'URL suivante qui décrit brièvement le projet : `http ://www.industrie.gouv.fr/rntl/AAP2001/Fiches_Resume/DICO.htm`.

2.6.4 Vis-à-vis de la granularité de l'analyse

Jusqu'à la fin des années 90, nous avions toujours testé nos approches sur des analyses par lot [Mé93, Mé94, Mé98, KNMH00, MMVM00]. En effet, cela nous semblait suffisant pour valider les concepts avancés. Depuis, nous avons révisé notre jugement. Ainsi, G^{N}G travaille en continu. De même, le prototype implantant l'approche présentée dans [ZMB02, ZMB03a, ZMB03b] réalise une analyse en continu.

2.6.5 Vis-à-vis du comportement après détection

L'objectif de nos travaux est de concevoir et de valider des méthodes de détection, pas de concevoir des systèmes totalement opérationnels. Aussi, nous nous sommes toujours contentés d'adopter une approche passive, consistant à informer l'opérateur de la détection d'une attaque. En outre, répondre automatiquement aux alertes nous parait aujourd'hui dangereux, tant la qualité des dites alertes nous semble insuffisante.

Nous n'avons donc aucune contribution dans le domaine des réponses actives, si ce n'est un élément de ADeLe [MM01], langage dans lequel une section permet de décrire la réponse à apporter en cas d'occurrence de l'attaque décrite.

Conclusion

Nous avons présenté dans ce chapitre les grandes caractéristiques des systèmes de détection d'intrusions, puis nous avons positionné nos travaux relativement à ces caractéristiques.

Nous avons contribué presque sur chaque axe possible, à l'exception notable de l'approche par scénarios exploitant des systèmes à base de règles. En effet, si cette technique peut être efficace appliquée à la corrélation d'alertes, elle nous semble mal adaptée à la corrélation d'événements, domaine que nous avons principalement exploré.

A l'issue de ces travaux, nous avons acquis la conviction que les systèmes de détection d'intrusions, quelque soit leurs caractéristiques, souffrent de leur peu de prise en compte (si ce n'est absence de prise en compte) de l'environnement dans lequel ils évoluent. Une telle prise en compte fait l'objet du chapitre 3 de ce mémoire. Les travaux de [ZMB02, ZMB03a, ZMB03b] et [MMDD02], peut abordés jusqu'à présent, y sont détaillés.

Chapitre 3

Prise en compte des caractéristiques du système surveillé et de sa politique de sécurité

3.1 Motivations

Mes activités dans le domaine de la détection d'intrusions se sont traduites depuis une dizaine d'années, d'une part par l'exploration de divers mécanismes (comportementaux ou à base de scénarios), d'autre part par l'application de ces mécanismes à divers environnements : machines Unix, CORBA, réseaux ad hoc.

Ces travaux ont été positionnés dans le domaine et décrits rapidement dans le chapitre 2. Ils se poursuivent actuellement, au travers de contrats de recherche et au travers de thèses que je co-dirige : celle de Cédric Michel (Langage de description d'attaque pour la détection d'intrusions en environnement réseau hétérogène, sous la direction de Gerardo Rubino, Université de Rennes 1) ; celle de Jean-Marc Percher (Sécurité dans les réseaux ad-hoc, sous la direction de Jean-Pierre Claudé, Université de Versailles) ; celle de Ricardo Puttini (Détection par approche comportementale des attaques contre les protocoles de routage ad hoc, sous la direction de Rafael de Sousa, Université de Brasilia) ; celle, enfin, d'Elvis Tombini (Filtrage comportemental puis analyse par scénarios d'attaques contre serveurs web, sous la direction de Mireille Ducassé, INSA de Rennes, et la codirection de Hervé Debar, France Télécom).

Ces travaux, quoique très utiles et, je crois, intéressants, s'inscrivent dans la continuité des approches classiques de la détection d'intrusions. Pourtant, bien qu'assez largement étudiées depuis 20 ans, ces approches classiques tardent à prou-

Année	1988	1989	1990	1991	1992
Incidents	6	132	252	406	773
Année	1993	1994	1995	1996	1997
Incidents	1334	2340	2412	2573	2134
Année	1998	1999	2000	2001	2002
Incidents	3734	9859	21756	52658	82094

TAB. 3.1 – Nombre d'incidents rapportés au CERT/CC chaque année depuis sa création (voir `www.cert.org`). Un incident peut impliquer un ou de nombreux sites.

ver leur efficacité opérationnelle. Les intrusions ne sont-elles pas toujours de plus en plus nombreuses ? Le nombre d'incidents de sécurité rapportés annuellement au CERT/CC (voir tableau 3.1) n'incite pas à l'optimisme.

Dès lors, que manque-t-il aux IDS actuels ? A mon sens, une information tout à fait fondamentale : la vision de l'environnement dans lequel il sont placés.

Dans ce chapitre, j'ai donc choisi de présenter, parmi mes travaux en cours, ceux qui tentent d'apporter une telle vision, tant au niveau du manager qu'à celui des sondes. Ces travaux se déroulent dans le cadre de deux thèses : celle de Benjamin Morin (*Gestion et corrélation des alertes issues des outils de détection d'intrusions*), menée sous la direction de Mireille Ducassé, INSA de Rennes, et la codirection de Hervé Debar (partenariat CIFRE entre Supélec et France Télécom) ; celle de Jacob Zimmerman (*Détection d'intrusions orientée politique basée sur un modèle de contrôle de flux d'information*), menée sous la direction de Gerardo Rubino, Université de Rennes 1.

La corrélation d'alerte au niveau des managers me semble indispensable à la résolution de certains des problèmes actuels de la détection d'intrusions, tels que le taux élevé de faux positifs ou la pauvreté du diagnostic porté par les alertes. Pour cela, le mécanisme de corrélation doit être capable de tenir compte des caractéristiques du système d'information surveillé (topologie, type de machines et de services, failles connues, politique de sécurité). C'est tout l'objet du travail que nous conduisons autour du modèle M2D2 et de son exploitation. Ce travail est présenté dans la première partie de ce chapitre.

Au niveau des sondes également, le contexte est important. Parmi les éléments de contexte, ceux liés à la politique de sécurité me semble essentielle. En effet, et de manière assez paradoxalement alors que la définition accepté par tous du terme « intrusion » est « toute violation de politique de sécurité », les sondes de détections actuelles ne tiennent aucunement compte de cette politique. Je pense que c'est une erreur. En outre, ma conviction est que c'est au niveau du système d'exploitation et au moment où les violations de politique se produisent, qu'il faut détecter les dites

violations, et éventuellement les empêcher (concept de *intrusion prevention*). Ces deux observations ont motivé le travail sur les flux de références, présenté dans la seconde partie de ce chapitre.

3.2 Corrélation d'alertes avec prise en compte de l'environnement surveillé

3.2.1 Objectifs de l'étude

Le problème actuel majeur des systèmes de détection d'intrusions est sans conteste le très grand nombre d'alertes générées. La plupart de ces alertes sont en fait des faux positifs (jusqu'à 90% dans certains cas, selon [LFG+00, DCW+99]). Comme se pose de surcroît le problème des faux négatifs, on est amené dans la pratique à multiplier les sondes, et donc à augmenter encore la quantité d'alertes produites.

Dans le cas où l'alerte est un vrai positif, un autre problème important se pose : le diagnostic offert par l'IDS est de trop faible qualité, les informations portées par les alertes étant insuffisantes. Cela oblige l'administrateur de sécurité à se plonger dans les données d'audit pour analyser finement ce qui s'est passé. L'alerte peut ainsi correspondre à l'exploitation d'une vulnérabilité qui n'est pas présente sur le système d'information attaqué. Si dans certains cas de telles tentatives infructueuses d'attaque doivent être signalées, dans d'autres cas on préfèrera au contraire les ignorer. Cette dernière stratégie est généralement impossible avec les IDS actuels.

Enfin, l'administrateur de sécurité a aussi besoin d'une vision « dynamique » de la situation dans laquelle se trouve le système d'information qu'il a sous surveillance. Ainsi, il est souhaitable de lui faire savoir en une et une seule alerte que s'est produit un scan de ports, suivi d'une série d'attaques contre le serveur web dont la dernière a été positive, suivi de la modification de la page d'accueil de ce serveur. En d'autres termes, si l'on se réfère aux définition de la page 6, on souhaite passer d'un système de détection d'intrusions à un système de détection dr'incidents de sÉCurité.

Nous nous intéressons à la résolution de ces trois problèmes. Dans les trois cas, une solution possible réside dans la corrélation des alertes émises par les sondes.

Dans ce contexte, le travail de thèse de Benjamin Morin vise à identifier l'ensemble des règles de corrélation utiles à la réduction du nombre d'alertes et à la diminution du taux de faux positifs, à l'amélioration du diagnostic porté par les alertes et à la détection d'incidents. En conséquence, son travail vise aussi à définir et à structurer l'ensemble des informations qu'il faudra posséder pour mettre en œuvre les différentes règles de corrélation identifiées.

Dans la suite, nous précisons ce que nous entendons par corrélation d'alertes, puis nous décrivons les informations que l'on doit avoir en possession pour implanter des formes de corrélation évoluées. Nous détaillons alors M2D2, un modèle formel ensembliste qui prend en compte une partie de ces informations. Enfin, avant de présenter les avantages et limites de notre approche et de conclure, nous donnons

deux exemples de corrélations exploitant des informations contenues dans M2D2.

3.2.2 Différentes formes de corrélation

La référence [JW93] définit la corrélation d'alertes comme « l'interprétation d'alertes multiples dans le but de leur attribuer de nouvelles significations ». Si l'on adopte cette définition, très générale, de nombreux travaux se sont intéressés à la corrélation d'alertes [OJC97, WZY+99, VS01, DW01, KTK01, DC01, Cup01, NRC01, GHH+01, GG01, CM02, NC02, GG02]. Cependant, le terme « corrélation » est générique et il recouvre en fait des mécanismes plus ou moins complexes. Malheureusement, la communauté ne s'est pas encore accordée sur une terminologie commune pour ses mécanismes.

Cuppens définit dans [Cup01] :
- l'*agrégation* : création d'un groupe d'alertes contenant toutes les alertes produites par une ou plusieurs sondes face à une même attaque ;
- la *fusion* : génération d'une nouvelle alerte contenant toutes les informations contenues dans les alertes d'un groupe ;
- la *corrélation* : regroupement des alertes résultant de l'agrégation, pour fournir à l'administrateur une information plus synthétique.

Nous ne retenons pas cette terminologie et ces définitions. Pour nous, l'agrégation est en effet une forme de corrélation. En outre, cette définition d'agrégation se limite à la constitution de groupes d'alertes produites par la même attaque. Or on peut aussi constituer des groupes suivant d'autres critères. En fait, ces définitions résultent du contexte dans lequel elles ont été produites, le projet MIRADOR, dans lequel il s'agissait de détecter des scénarios d'attaque. Or, ce n'est pas là le seul objectif que nous voyons à la corrélation.

Nous proposons ici la typologie suivante des mécanismes de corrélation :
- la *compression* : réduction des multiples occurrences de la même alerte en une seule ; le *décompte* est une forme particulière de compression dans laquelle on réduit un nombre spécifié d'occurrences ;

- la *déduction* : la substitution d'un ensemble d'alertes satisfaisant une certaine condition par une nouvelle alerte.
  On retrouve pour la condition à satisfaire les deux types de contraintes déjà définis pour les événements (voir paragraphe 2.2.2, approche orienté langage) :
  - les contraintes logiques qui définissent des contraintes sur les valeurs portées par les champs des différentes alertes ;
  - les contraintes temporelles qui elles définissent un ordre total ou partiel que doivent respecter les alertes.

  On peut se contenter de spécifier une contrainte simple, par exemple portant sur l'identité de l'identifiant de l'attaque rapportée par les différentes alertes. Dans ce cas, on parle d'*agrégation* d'alertes, comme [Cup01].

A contrario, l'ensemble des contraintes peut définir une signature d'attaque complexe. Dans ce cas, la corrélation est une forme d'approche par scénarios, mise en œuvre dans le manager et non plus dans les sondes.
La nouvelle alerte produite est obtenue par *fusion*, c'est-à-dire en reprenant toutes les informations portées par les alertes de l'ensemble satisfaisant la condition ;

- la *généralisation* : lorsqu'une hiérarchie d'alertes a été définie, la généralisation consiste à remplacer une alerte par sa super-classe. Ainsi, à titre d'exemple, on peut souhaiter remplacer une alerte indiquant une attaque en provenance d'une adresse IP particulière, par une alerte similaire mais indiquant le fournisseur d'accès de l'attaquant ; dans ce cas, la généralisation porte sur l'adresse IP source de l'attaque. En un certain sens, on a là une forme de déduction par contrainte logique. Cependant, nous préférons parler de généralisation, car la contrainte ne s'exprime pas par des comparaisons entre valeurs de certains champs de plusieurs alertes ; elle porte au contraire sur la valeur d'un champ donné dans une alerte donné ;

- la *suppression* : lorsque les alertes possèdent des niveaux de criticité, la suppression consiste à inhiber les alertes de faible criticité en présence d'alertes de forte criticité.

L'ensemble de ces mécanismes peut permettre de réduire le nombre d'alertes. La réduction du taux de faux positifs, l'amélioration du diagnostic et la détection d'incidents reposent quant à elles essentiellement sur la déduction et la généralisation. Les contraintes logiques de la déduction et les hiérarchies de la généralisation portent sur des connaissances qui font l'objet du paragraphe suivant.

3.2.3 Connaissances nécessaires à la déduction

Quatre types de connaissances sont selon nous nécessaires à la déduction : celles relatives aux attaques et aux alertes, celles relatives au système d'information surveillé et à ses vulnérabilités, celles relatives aux sondes utilisées et celles relatives à la politique de sécurité en place sur le système surveillé. Nous donnons quelques détail dans la suite de ce paragraphe.

— *Connaissances sur les attaques et les alertes*

A notre connaissance, les travaux conduits à ce jour en corrélation d'alertes exploitent des informations contenues, d'une part dans des bases de signatures d'attaque, d'autre part dans les alertes à corréler.

Il en est ainsi de l'IDS distribué G^NG, présenté brièvement dans le paragraphe 2.6.3. G^NG propose une forme particulière de déduction qui lie entre elles les alertes provenant d'une ou plusieurs sondes et correspondant à une signature d'attaque donnée, exprimée sous forme de règles logiques ne portant que sur de l'information contenue dans les alertes.

— *Connaissances sur le système d'information surveillé et sur ses vulnérabilités*

Comme nous l'avons déjà mentionné, dans le cas où une alerte correspond à l'exploitation d'une vulnérabilité absente du système d'information, on peut souhaiter l'ignorer. A cette fin, il faut connaître, d'une part les éléments matériels et logiciels composant ce système d'information (SI), d'autre part les vulnérabilités connues de ces éléments.

Le composant matériel le plus important du SI est bien sûr l'ordinateur, sur lequel fonctionne un ensemble de logiciels : un système d'exploitation (dont il faudra tenir compte du type, de la version et des correctifs éventuels) et un ensemble de services et d'applications (dont il faudra également tenir compte des versions). Un ordinateur présente des vulnérabilités liées, soit à un élément particulier, soit à une combinaison particulière d'éléments. D'autres types de matériel sont aussi à prendre en compte, comme les équipements réseau.

L'expérience montre que certains types de machines (ensemble matériels et logiciels) provoquent parfois, dans des circonstances précises, des flots d'alertes en absence d'attaque. Il est bien sûr utile d'ignorer ces alertes. Il faut donc avoir connaissance de ces types de machine.

Les composants matériels sont organisés en réseau. La topologie de ce réseau doit être connue, afin de permettre la corrélation d'alertes résultant de la propagation, sur les différents composants du SI, de la même attaque (on pense bien sûr aux vers) ou du même attaquant (qui peut enchaîner des attaques différentes, en fonction du contexte qu'il rencontre). Dans ce dernier cas, il s'agit de détection d'incidents.

Notons pour finir que de cette connaissance des composants du SI, peut en outre résulter une identification précise des cibles d'attaque.

— *Connaissances sur les sondes utilisées*

Dans un environnement multi-sondes, il faut être capable de comprendre pourquoi une sonde a émis une alerte quand une autre n'a rien fait. Il faut donc savoir quels composants du SI est sous la surveillance de quelles sondes, car il est normal qu'une sonde ne réagisse pas si les manifestations de l'attaque ne lui sont pas visibles.

Par ailleurs, certains IDS génèrent des flots d'alertes en présence d'une attaque donnée. Cette caractéristique doit être connue, car il peut être utile de regrouper ces alertes en une seule.

— *Connaissances sur la politique de sécurité en place sur le système d'information surveillé*

Conformément à la définition donnée page 6, une intrusion est une violation de la politique de sécurité. Il serait donc bien sûr intéressant de confronter le flux d'alertes à la politique, afin, par exemple, d'éliminer les alertes correspondant à des situations autorisées.

Les mesures de sécurité en place sur le site surveillé peuvent prévoir des tests de pénétration, périodiques ou apériodiques. Ces tests vont bien sûr générer de nombreuses alertes. Il est souhaitable de savoir qu'un tel test est en cours, pour s'assurer que les IDS le détectent, mais pour ne pas tenir compte des alertes correspondantes.

3.2.4 M2D2, un modèle formel pour la corrélation d'alertes

La première contribution du travail de thèse de Benjamin Morin est un modèle formel de données regroupant les informations utiles aux différentes formes de corrélation. Ce modèle, baptisé M2D2, prend pour le moment en compte quatre types d'information (les travaux actuels en corrélation ne tiennent généralement compte que du dernier type) :

1. les composants et la topologie du SI ;
2. les vulnérabilités connues de ces composants ;
3. les outils de sécurité passive (IDS et scanner de vulnérabilité) mis en place sur le SI ;
4. les actions effectuées sur le SI.

M2D2 est un modèle ensembliste. Les ensembles définis et les relations et fonctions définies sur ces ensembles sont présentés par la figure 3.1. Ce paragraphe présente les éléments du modèle[1] ; le paragraphe suivant en donne trois exemples d'exploitation.

Nous utilisons ici les notions de relation, fonction totale et fonction partielle [Abr96] pour décrire les cardinalités :
- une relation 'etablit une correspondance de cardinalités (0,N)-(0,N) :
- une fonction partielle établit une correspondance de cardinalités (0,N)-(0,1) ;
- une fonction totale établit une correspondance de cardinalités (1,N)-(0,1).

— *Les composants et la topologie du SI*

M2D2 considère pour le moment que les composants du SI sont des ordinateurs (*host*). L'ensemble des hosts est noté $\mathcal{H}$.

Sur chaque host résident des logiciels. Chaque logiciel appartient à l'ensemble $\mathcal{P}$ des produits. La fonction **configuration** définit le sous-ensemble de $\mathcal{P}$ qui réside sur chaque host (nous appelons « configuration » ce sous-ensemble). **configuration** est une fonction définie de $\mathcal{H}$ vers $\mathrm{P}(\mathcal{P})$, ensemble de tous les sous-ensemble de $\mathcal{P}$.

Un produit en lui-même est défini par un quadruplet : (vendeur, nom, version, type). Nous maintenons des ensembles VN, PN, PV et PT de noms de vendeurs, de noms de produits, de numéros de version et de types de produit (système d'exploitation, application locale, service réseau). Les fonctions **vendor** (partielle), **prodname**

[1] Nous donnons ici une description informelle de M2D2. La description formelle est donnée dans [MMDD02]

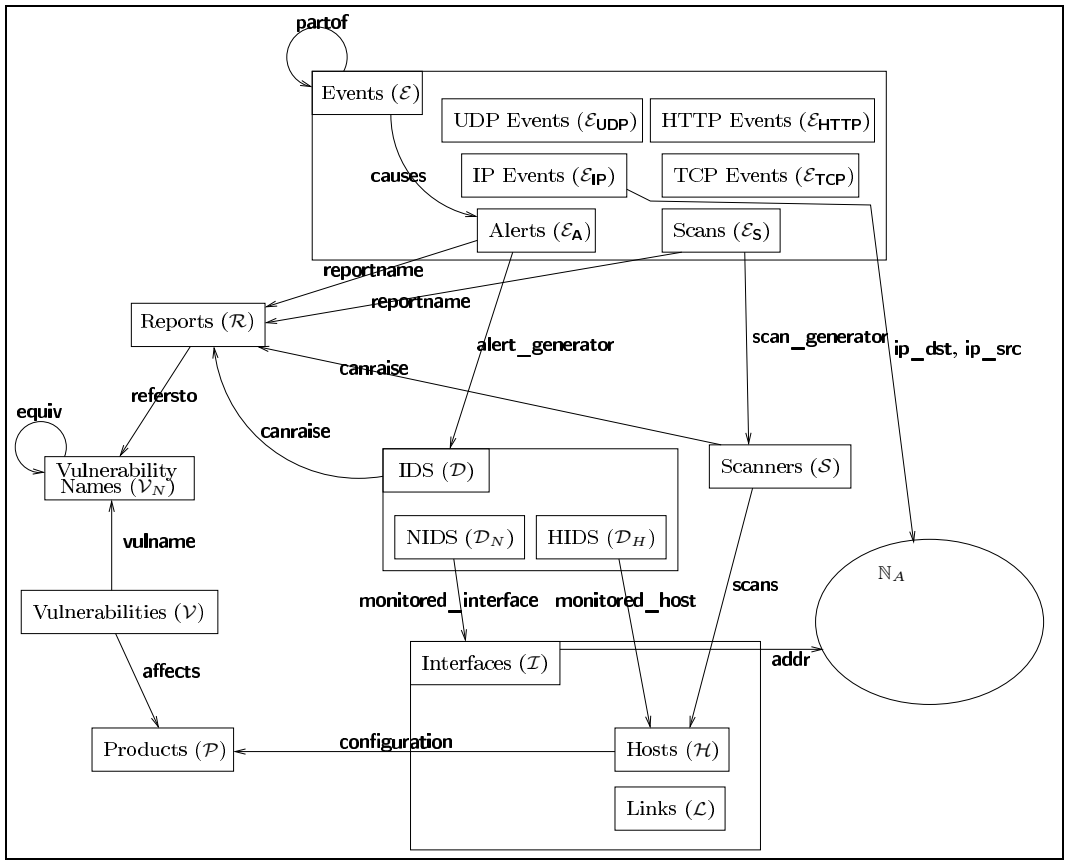


FIG. 3.1 – Le modèle M2D2 prend en compte 4 types d'information : les composants et la topologie du SI (interfaces, hosts, links) ; les vulnérabilités connues de ces composants ; les outils de sécurité passive (IDS et scanner de vulnérabilité) mis en place sur le SI ; les actions effectuées sur le SI (events de différents types, dont alerts).

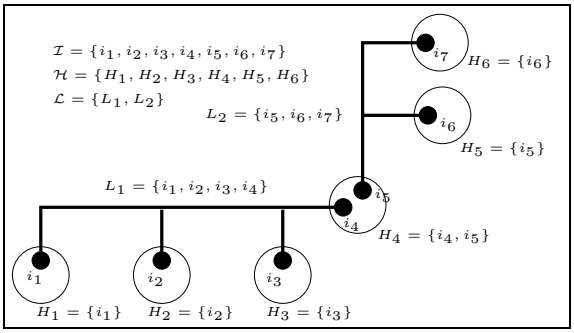


FIG. 3.2 – Modèle de réseau d'après [Vig96]

(totale), **version** (partielle) et **prodtype** (partielle) sont définies respectivement de $\mathcal{P}$ vers VN, PN, PV et PT.

Pour la topologie, nous avons repris un modèle proposé par [Vig96]. Chaque host peut contenir plusieurs interfaces réseau. Les interfaces des différents hosts sont reliés entre eux pour former des liens (*link*). Les ensembles d'interfaces et de liens sont respectivement notés $\mathcal{I}$ et $\mathcal{L}$. Cette modélisation est illustrée par la figure 3.2.

Une fonction injective (chaque interface ne peut avoir qu'une adresse), **addr**, est définie de $\mathcal{I}$ vers l'ensemble $\mathbb{N}_A$ des adresses IP. Pour le moment, M2D2 ne tient pas compte de la dynamicité possible des adresses IP (cas de DHCP). Il le fera à l'avenir.

Une fonction totale **hostname** est définie de $\mathcal{H}$ vers l'ensemble des noms de machine du SI. En effet, dans certains cas, ce nom figure dans les alertes. Dans d'autres cas, ce sont des adresses IP qui y figurent. La fonction **hostname** permet de faire le lien.

La modélisation du SI dans M2D2 doit être enrichie. En particulier, la version actuelle ne permet pas de représenter les notions importantes que sont utilisateurs, processus ou fichiers. En conséquence, la politique d'authentification et de droits d'accès d'une machine donnée n'est pas modélisable pour le moment.

— *Les vulnérabilités connues des composants du SI*

Nous prenons en compte l'ensemble des vulnérabilités rendues publiques par la base ICAT[2]. Nous appelons $\mathcal{V}$ cet ensemble. Ces vulnérabilités sont nommées d'après la convention CVE/CAN[3]. Ces noms appartiennent à l'ensemble $\mathcal{V}_N$. Une fonction injective **vulname** est définie de $\mathcal{V}$ vers $\mathcal{V}_N$.

[2] `http ://icat.nist.gov`
[3] `http ://cve.mitre.org`

D'autres systèmes de nommage sont aussi utilisés (ISS X-Force, CyberCop Scanner, CERT, etc.). Les noms correspondants appartiennent également à $\mathcal{V}_N$. Une liste d'équivalence entre les noms CVE et ces autres noms est proposée par le MITRE. Cette liste est modélisée dans M2D2 au moyen d'une relation et non d'une fonction. Nous appelons **equiv** cette relation.

Conformément à [Shi00], une vulnérabilité est définie comme une faille dans la conception, l'implémentation ou l'administration d'un logiciel ou d'une combinaison de logiciels utilisés sur un host du SI, cette faille pouvant être exploitée pour violer la politique de sécurité en vigueur. Cette définition est traduite par la relation **affects**, définie de $\mathcal{V}$ vers $\mathrm{P}(\mathcal{P})$.

Nous souhaitons aussi spécifier dans M2D2 certaines caractéristiques liées à l'exploitation des vulnérabilités, car ces caractéristiques peuvent être exploitées par la corrélation. Ainsi, dans certains cas, on peut souhaiter tenir compte de l'éventuelle augmentation du niveau de privilège d'un processus utilisateur.

Pour le moment, nous avons choisi deux caractéristiques essentielles : le type d'accès initial au SI que demande l'exploitation de la vulnérabilité et les conséquences de cette exploitation.

Les types d'accès que nous retenons sont :

- distant : l'exploitation de la vulnérabilité demande que l'attaquant puisse joindre le host attaqué par le réseau ;
- utilisateur : l'exploitation de la vulnérabilité demande la possibilité de se connecter en tant qu'utilisateur (localement ou à distance) sur le host attaqué ;
- physique : l'exploitation de la vulnérabilité demande un accès physique au host attaqué.

Les conséquences que nous retenons sont :

- déni de service : l'exploitation de la vulnérabilité rend indisponible une ressource du host attaqué pour ses utilisateurs légitimes ;
- divulgation d'information : l'exploitation de la vulnérabilité résulte en l'accès illégal à de l'information (contenu d'un fichier, d'un répertoire, etc.) ;
- exécution de code : l'exploitation de la vulnérabilité conduit à l'exécution d'un programme que l'attaquant ne peut normalement utiliser (c'est le cas des attaques de types *buffer overflow* ou *race condition*).

Nous maintenons un ensemble des types d'accès et un ensemble des types de conséquences. Deux fonctions totales **req** et **con** sont définies de $\mathcal{V}$ vers ces deux ensembles.

— *Les outils de sécurité passive en place*

La version actuelle du modèle prend en compte les outils permettant de détecter les failles de sécurité (scanners) et les outils permettant de détecter l'exploitation de ces failles (IDS). L'ensemble des scanners utilisés sur le SI est appelé $\mathcal{S}$; celui des IDS, $\mathcal{D}$.

Ce dernier ensemble est constitué par l'union du sous-ensemble $\mathcal{D}_N$ des IDS à sonde réseau (NIDS) et de celui $\mathcal{D}_H$ des IDS à sondes système ou applicatives (HIDS). La fonction **data** est définie de $\mathcal{D}$ vers l'ensemble {HIDS, NIDS}. Son inverse permet de construire les ensembles $\mathcal{D}_N$ et $\mathcal{D}_H$.

Cette distinction classique a un lien fort avec ce que nous appelons la « *visibilité topologique* » d'un IDS, c'est-à-dire le sous-ensemble des composants du SI que cet IDS a sous surveillance. La visibilité topologique d'un HIDS se limite généralement à la machine sur lequel est installée la sonde système ou applicative, celle d'un NIDS s'étend à toutes les machines dont les échanges passent par le lien réseau sur lequel la sonde réseau est installée. Cette caractéristique est modélisée par les fonctions **monitored_host** et **monitored_interface** définies respectivement de $\mathcal{D}_H$ vers $\mathcal{H}$ (ensemble des hosts) et de $\mathcal{D}_N$ vers $\mathcal{I}$ (ensemble des interfaces). La visibilité topologique d'un scanner est donnée par la liste des machines qu'il scanne. La relation **scans** est définie de $\mathcal{S}$ vers $\mathcal{H}$ à cet effet.

La connaissance de la visibilité topologique permettra au module de corrélation d'alertes de déterminer si un IDS peut ou non réagir à une attaque détectée par un autre IDS. Si cette réponse est positive, il peut y avoir suspicion de faux positif de la part de cet autre IDS.

IDS et scanners génèrent des alertes. Chacune est identifiée par un nom (appelé ici nom de rapport), propre à l'outil qui l'a généré. Nous appelons $\mathcal{R}$ l'ensemble des noms de rapport de tous les IDS et scanners en fonction dans le SI. La relation **canraise** est définie de $\mathcal{D} \cup \mathcal{S}$ vers $\mathcal{R}$, afin de définir quelle alerte peut être générée par quel IDS ou quel scanner. Dans le cas d'un IDS par scénarios, il est possible de déterminer (par examen des scénarios) les vulnérabilités exploitées par chaque scénario. Ceci est modélisé par la relation **refersto**, définie de $\mathcal{R}$ (pour le moment, M2D2 ne permet pas de modéliser les scénarios ; cependant, à chaque scénario correspond un nom de rapport ; $\mathcal{R}$ est donc l'ensemble de départ) vers $\mathcal{V}$, l'ensemble de vulnérabilités.

La composition de **canraise** et **refersto** permet de déterminer l'ensemble des vulnérabilités dont l'exploitation peut être détectée par un IDS donné. C'est ce que nous appelons la « *visibilité opérationnelle* » de l'IDS. Au même titre que la visibilité topologique, c'est une caractéristique intéressante, car elle permet d'identifier d'éventuels faux positifs.

Bien évidemment, est modélisé le type de détection (comportemental ou par scénarios) mis en œuvre par l'IDS. La fonction **meth** est définie dans ce but de $\mathcal{D}$ vers l'ensemble {Comportemental, scénarios}. Pour autant, cette information n'est pour le moment pas exploitée par les corrélations que nous avons spécifiées.

— *Les actions effectuées sur le SI*

Conformément aux définitions données dans le chapitre d'introduction de ce mémoire, les sources de données génèrent de l'information sur les activités des entités

du SI. Ces données sont ensuite filtrées et formatées par un capteur, pour former des événements. Ce sont ces événements, traduction des actions effectuées sur le SI, qui sont modélisés dans M2D2. Nous avons en outre noté dans ce même chapitre d'introduction, que le modèle présenté par la figure 1.1 est par nature récursif, un manager pouvant être considéré par un capteur comme une source de données. Pour être en accord avec cette caractéristique, les alertes et les événements forment une même entité de M2D2, les *events*. L'ensemble des events est appelé $\mathcal{E}$.

Pour le moment, nous avons restreint les types d'événements pris en compte dans le modèle à ceux qui sont les plus courants pour les IDS commerciaux actuels. Nous considérons donc que $\mathcal{E}$ est l'union des sous-ensembles suivants : $\mathcal{E}_{\mathbf{A}}$ (ensemble des alertes), $\mathcal{E}_{\mathbf{S}}$ (ensemble des scans), $\mathcal{E}_{\mathbf{IP}}$ (ensemble des trames IP), $\mathcal{E}_{\mathbf{UDP}}$ (ensemble des paquets UDP), $\mathcal{E}_{\mathbf{TCP}}$ (ensemble des paquets TCP), $\mathcal{E}_{\mathbf{HTTP}}$ (ensemble des requêtes HTTP), $\mathcal{E}_{\mathbf{LOG}}$ (ensemble des lignes du journal des accès à un serveur web). $\mathcal{E}_{\mathbf{A}}$, $\mathcal{E}_{\mathbf{S}}$, $\mathcal{E}_{\mathbf{IP}}$, $\mathcal{E}_{\mathbf{UDP}}$, $\mathcal{E}_{\mathbf{TCP}}$, $\mathcal{E}_{\mathbf{HTTP}}$et $\mathcal{E}_{\mathbf{LOG}}$ partitionnent $\mathcal{E}$. A l'avenir d'autres types d'événements devront être pris en compte, comme par exemple les lignes de log système.

Certains événements en encapsulent d'autres. Ainsi, une ligne du journal des accès à un serveur web encapsule une requête HTTP ; un paquet UDP encapsule des trames IP ; une alerte peut résulter de la présence d'autres alertes et les encapsuler ; etc. Cette caractéristique est modélisée par la relation **partof**, définie de $\mathcal{E}$ vers lui même.

Chaque type d'événement porte des attributs différents. Ainsi, seuls les événements TCP et UDP portent les attributs port source et port destination. Le seul attribut commun à tous les types d'événement est une estampille temporelle. Une fonction **tstamp** est ainsi définie de $\mathcal{E}$ vers $\mathbb{N}$.

Dans la suite de ce paragraphe, nous détaillons les autres fonctions définies pour les scans et les alertes. Les fonctions définies pour les autres types d'événements ne sont pas détaillées ici (le lecteur intéressé se reportera à [MMDD02]).

Nous avons déjà mentionné que chaque alerte est identifiée par un nom de rapport et que l'ensemble $\mathcal{R}$ est l'ensemble des noms de rapport. La relation **reportname** est définie de $\mathcal{E}_{\mathbf{A}} \cup \mathcal{E}_{\mathbf{S}}$ vers $\mathcal{R}$.

Chaque alerte ou scan est généré par un outil donné : ainsi, sont définies deux fonctions **alert_generator** et **scan_generator** respectivement de $\mathcal{E}_{\mathbf{A}}$ vers $\mathcal{D}$ (ensemble des IDS) et de $\mathcal{E}_{\mathbf{S}}$ vers $\mathcal{S}$(ensemble des scanners).

Les alertes sont causées par des événements qui sont des manifestations d'attaque. La relation **causes** est définie de $\mathcal{E}$ vers $\mathcal{E}_{\mathbf{A}}$ pour en rendre compte. Il faut noter qu'une alerte peut ne pas posséder d'événement causal (car cet événement n'est peut-être pas disponible), qu'une alerte peut posséder plusieurs événements causaux (cas de la reconnaissance de scénarios) et qu'un même événement peut être cause de plusieurs alertes (par exemple une requête HTTP contenant plusieurs patterns « douteux » générera plusieurs alertes).

Nous ne définissons pas de fonction pour modéliser la source et la cible de l'attaque ayant généré l'alerte. En effet, cette information est portée par les événements causaux (par exemple adresses IP source et destination). En revanche, pour les scans, qui ne sont pas générés par des événements, nous définissons les fonctions **scan_host_target** et **scan_port_target** respectivement de $\mathcal{H}$ (ensemble des hosts) et $\mathbb{N}$, pour modéliser le host cible du scan et le port visé.

3.2.5 Exemples d'exploitation de M2D2 en corrélation d'alertes : identification des faux positifs

Nous illustrons ici l'exploitation de M2D2 par deux exemples de corrélations visant à réduire le nombre d'alertes et le taux de faux positifs : l'élimination des alertes générées par des attaques tentant d'exploiter des vulnérabilités absentes des composants du SI et l'élimination des alertes générées par un IDS suite à une attaque qui reste non détectée par d'autres IDS pourtant à même de la détecter.

— *Elimination des alertes générées par des attaques exploitant des vulnérabilités absentes des composants du SI*

Soit a l'alerte à laquelle on s'intéresse. L'ensemble des vulnérabilités exploitées par l'attaque ayant généré a, est donné par :

$$\mathcal{V}_a = \textbf{vulname}^{-1}[\textbf{equiv}[\textbf{refersto}(\textbf{reportname}(a))]]$$

Ces vulnérabilités affectent l'ensemble des configurations suivantes :

$$\mathcal{C}_a = \textbf{affects}[\mathcal{V}_a]$$

L'ensemble des configurations présentes sur le SI est donné par l'ensemble des antécédents de la fonction **configuration**. On le note $ran(\textbf{configuration})$.

L'attaque peut réussir si au moins un host possède une configuration qui le rend affectable. En d'autres termes, elle réussit si :

$$ran(\textbf{configuration}) \cap \mathcal{C}_a \neq \emptyset$$

Si cette condition n'est pas vérifiée, l'alerte peut être ignorée. Dans le cas contraire, l'ensemble de toutes les machines vulnérables est obtenu par :

$$\textbf{configuration}^{-1}[ran(\textbf{configuration}) \cap \mathcal{C}_a]$$

— *Elimination des alertes générées par un IDS suite à une attaque qui reste non détectée par d'autres IDS capables de la détecter*

Soit a l'alerte à laquelle on s'intéresse. L'ensemble $\mathcal{V}_a$ des vulnérabilités exploitées par l'attaque ayant généré a, est obtenu comme précédemment.

Il s'agit de déterminer, dans un premier temps l'ensemble des IDS dont les vul-

nérabilités exploitées par a appartiennent à la fois à la visibilité opérationnelle et à la visibilité topologique, puis, dans un second temps, si ces IDS ont émis une alerte similaire à a.

L'ensemble des IDS dont les vulnérabilités exploitées par a appartiennent à la visibilité opérationnelle, est donné par :

$$\mathcal{D}_a^{oper} = \textbf{canraise}^{-1}[\textbf{refersto}^{-1}[\mathcal{V}_a]]$$

La détermination de l'ensemble des IDS dont les vulnérabilités exploitées par a appartiennent à la visibilité topologique, pose quant à elle plusieurs problèmes, liés aux NIDS. Savoir quels NIDS auraient dû réagir implique de connaître la route suivie par les paquets réseau, ce qui n'est généralement pas le cas. En outre, lorsque l'alerte à laquelle on s'intéresse a été générée par un HIDS, comment savoir si les NIDS étaient à même de détecter l'attaque correspondante. En effet, cette attaque peut être locale et n'impliquer aucun échange réseau.

Pour ces deux raisons, nous nous limitons donc à déterminer l'ensemble des HIDS dont les vulnérabilités exploitées par a appartiennent à la visibilité opérationnelle.

Si a a été générée par un HIDS, il faut déterminer l'ensemble des HIDS qui surveillent la même machine. Cet ensemble est donné par :

$$\mathcal{D}_a^{topo-1} = \textbf{monitored\_host}^{-1}[\textbf{monitored\_host}(\textbf{alert\_generator}(a))]$$

Si a a été générée par un NIDS, il faut déterminer l'ensemble des HIDS qui surveille la machine dont l'adresse IP est l'adresse destination des paquets qui ont déclenché l'alerte. La fonction **ip_dst** (non décrite dans ce mémoire, voir [MMDD02]) appliquée à une trame IP rend l'adresse destination de ce paquet. Cet ensemble est alors donné par :

$\mathcal{D}_a^{topo-2} = \textbf{monitored\_host}^{-1}[\textbf{belongs\_to\_host}[\textbf{addr}^{-1}(\textbf{ip\_dst}(\textbf{causes}^{-1}(a))]]$

expression dans laquelle **belongs_to_host** est une fonction permettant d'identifier de manière unique un host à partir de son interface réseau.

Finalement, l'ensemble des HIDS dont les vulnérabilités exploitées par a appartiennent à la visibilité opérationnelle est donné par :

$$\mathcal{D}_a^{topo} = \mathcal{D}_a^{topo-1} \cup \mathcal{D}_a^{topo-2}$$

et l'ensemble des HIDS dont les vulnérabilités exploitées par a appartiennent à la fois à la visibilité opérationnelle et à la visibilité topologique par :

$$\mathcal{D}_a = \mathcal{D}_a^{oper} \cup \mathcal{D}_a^{topo}$$

Si aucun (ou peu) des IDS de $\mathcal{D}_a$ n'a émis une alerte similaire à a, cette dernière peut être ignorée. Reste donc à définir ce qu'est une alerte similaire. Différentes approches sont envisageables. Par exemple, on déterminera si une alerte exploitant les mêmes vulnérabilités a été émise par un IDS de $\mathcal{D}_a$, dans un intervalle de temps donné encadrant **tstamp**(a).

L'ensemble des alertes générées par les IDS de $\mathcal{D}_a$ est donné par :

$$\mathcal{V}_{HIDS} = \textbf{equiv}[\textbf{refersto}[\textbf{reportname}[\textbf{alert\_generator}^{-1}[\mathcal{D}_a]]]]$$

Donc, si : $\mathbf{reportname}^{-1}[\mathbf{refersto}^{-1}[\mathcal{V}_{HIDS}]] = \emptyset$, a peut-être ignorée.

3.2.6 Avantages et limites

En appuyant sur M2D2 la corrélation d'alertes, on utilise avec profit les liens existants entre vulnérabilités et topologie, entre topologie et outils de sécurité et entre outils de sécurité et vulnérabilités. Il s'agit là selon nous d'informations essentielles à prendre en compte pour l'implantation de formes de corrélations évoluées.

En outre, les entités (ensemble, fonctions, relations) de M2D2 étant définies formellement **??**, les règles de corrélation peuvent être exprimées rigoureusement et sans ambiguïté.

Pour autant, M2D2 est encore incomplet. Il ne couvre qu'une partie des connaissances utiles à la corrélation. A l'avenir les informations suivantes devront être ajoutées : les signatures d'attaque, les types de machines et les circonstances dans lesquelles ils peuvent donner lieu à alerte(s) en l'absence d'attaque, le comportement des sondes face à des attaques particulières, la politique de sécurité en vigueur sur le SI surveillé.

En fait, la version actuelle de M2D2 met l'accent sur les vulnérabilités. Par exemple, on ne trouve pas de relation permettant de définir des équivalences entre nom de rapport. Nous avons montré dans les corrélations décrites que si le besoin se fait sentir, on passe par les équivalences entre vulnérabilités sous-jacentes aux alertes. Se pose alors un problème pour les alertes non liées à une vulnérabilité, comme celles émises par les IDS comportementaux.

3.2.7 Conclusion

Le travail présenté ici n'est qu'une étape de notre étude. Outre les informations qu'il faudra ajouter au modèle, doivent aussi être étudiés les moyens à déployer pour mettre en œuvre efficacement le modèle et les corrélations qui l'exploitent.

L'implémentation actuelle du modèle utilise à la fois une base de données relationnelle et des faits prolog. Les règles de corrélations (déduction) sont elles-mêmes exprimées en SQL et en prolog, en fonction des éléments du modèle exploités. En outre, la généralisation s'appuie sur des arbres de concepts. Une telle implantation sera-t-elle assez efficace pour traiter les alertes en temps réel ? Si c'est le cas, dans quelles limites ?

L'ensemble de ce travail a été réalisé dans le cadre de la thèse de Benjamin Morin, sous la direction de Mireille Ducassé (INSA de Rennes) et la co-direction conjointe de Hervé Debar (France Télécom R&D). Une communication a été faite à RAID en octobre 2002 [MMDD02]. Deux autres articles sont en cours de revue et préparation.

3.3 Détection d'intrusions orientée politique basée sur un modèle de contrôle de flux d'information

Tout système de détection d'intrusions a pour but de détecter les violations de la politique de sécurité en vigueur sur le site surveillé. Pourtant, paradoxalement, les IDS actuels ne prennent généralement pas en compte cette politique. Dans le paragraphe 3.2, nous avons mentionné que la politique devait être un des élements à prendre en compte dans le processus de corrélation. Dans ce paragraphe, nous présentons un modèle de détection implantable, non plus au niveau du manager, mais au niveau d'une sonde, au cœur du système d'exploitation.

3.3.1 Systèmes d'exploitation classiques et attaques par délégation

Considérons par exemple la politique de sécurité simple suivante :
- chaque utilisateur est authentifié par un mot de passe,
- chaque utilisateur dispose d'un accès illimité aux fichiers dont il est le propriétaire,
- chaque utilisateur peut modifier lui-même son mot de passe (ce qui implique la lecture et l'écriture dans le fichier de mots de passe).

Sur un système d'exploitation classique, tels que Unix ou Windows, implantant une telle politique par le biais d'un contrôle d'accès discrétionnaire, les accès au fichier des mots de passe se font par l'intermédiaire de programmes privilégiés, afin de restreindre ces accès aux seuls besoins d'authentification et de modification de mots de passe.

Faisons maintenant l'hypothèse :
- que l'implémentation du système d'exploitation en question soit correcte, c'est-à-dire qu'il soit effectivement impossible d'exécuter une action interdite (comme la lecture du fichier des mots de passe par un utilisateur sans passer par un programme privilégié) ;
- que l'implémentation des programmes privilégiés soit elle aussi correcte (c'est-à-dire, par exemple, qu'ils ne laissent pas une copie du fichier des mots de passe dans un répertoire d'accès libre pour un utilisateur quelconque) ;

Même sous cette hypothèse, un enchaînement astucieux de commandes système impliquant un ou plusieurs programmes privilégiés, peut potentiellement permettre à un utilisateur d'accèder à l'information contenue dans le fichier des mots de passe et donc de violer la politique de sécurité. Par exemple, sur un système Unix classique, l'utilisateur malicieux utilisera une attaque du type de celle de « l'impression symbolique » (voir figure 3.3).

C'est à ce type d'attaque, que l'on appelle *attaque par délégation*, que l'on s'intéresse dans cette étude. Il s'agit de suites d'actions, où chaque action est légale par

	actions de l'attaquant	actions du démon *lpr*
1	`lpr f1`	
2		vérification des droits sur `f1`
3	`rm f1`	
4	`ln -s /etc/shadow f1`	
5		lecture de `f1`, donc de `/etc/shadow`
6		envoi le contenu de `/etc/shadow` à l'imprimante

FIG. 3.3 – Attaque par délégation exploitant une faute de conception dans l'impression symbolique sous Unix. Cette attaque permet l'impression d'un fichier quelconque, même si l'attaquant ne possède pas de droits de lecture sur ce fichier. L'attaquant fait agir le démon *lpr* à sa place.

elle même, mais dont l'effet combiné viole la politique de sécurité. Dans la pratique, de très nombreuses attaques sont de cette catégorie, en particulier les attaques de type *race condition* ou *buffer overflow*. Cependant, il faut noter que l'on s'intéresse aux attaques violant les objectifs de sécurité en termes d'intégrité et de confidentialité des données ; le déni de service est en dehors de la portée de cette étude.

3.3.2 Détecter les attaques par délégation

Comme nous l'avons dit dans le paragraphe 3.1, les approches classiques de la détection d'intrusions peinent à faire preuve de leur efficacité opérationnelle. C'est particulièrement vrai sur les attaques par délégation. L'approche classique la plus à même de détecter ces attaques est l'approche par scénario. Cependant, même si l'on fait ici l'hypothèse que cette approche fonctionne correctement, encore faudra-t-il être capable de lui fournir les signatures des attaques qu'il lui faut détecter. Dans le cas qui nous préoccupe, ces signatures et leurs variantes sont très nombreuses et, sans doute, dans une large mesure inconnues. Nous avons donc écarté les approches classiques et souhaité explorer une autre voie.

Les attaques par délégation peuvent être décrites en termes de flux d'information. Ainsi, dans l'exemple de l'impression symbolique, on a un flux de `/etc/shadow` vers l'imprimante. Ces attaques peuvent donc théoriquement être traitées en utilisant des modèles de contrôle de flux d'informations. Contrairement aux modèles de contrôle d'accès qui ne régulent, par essence, que l'accès aux informations contenues dans les objets, les modèles de contrôle de flux (proposés initialement par [Den76] comme une extension du modèle de Bell et LaPadula [BL73]) régulent ce que les sujets[4] peuvent faire avec les informations auxquelles ils ont accèdé. Le contrôle de flux permet donc d'empêcher des transferts illégaux d'informations tels que le

[4]Pour faire simple dans le cadre de ce paragraphe, disons que les « objets » sont des fichiers et les « sujets » des processus. Une définition précise de ce que nous entendons par « objet » est donnée dans le paragraphe 3.3.5.

contenu du fichier des mots de passe.

Notre objectif est donc de proposer une approche de détection s'appuyant sur un modèle de contrôle de flux qui vérifiera la légalité des flux d'information entre objets du système, en fonction d'une politique de sécurité donnée. Notons cependant que les flux par canaux cachés ne sont pas gérés.

3.3.3 Cas du contrôle d'accès discrétionnaire

Nous souhaitons proposer un système applicable sur des machines classiques, de type Windows ou Unix, sur lesquelles tournent des applications dont on ne possède pas forcément le code source.

En conséquence, nous ne pouvons prendre en compte les opérations atomiques de chaque programme (affectation, addition, etc.) et nous nous intéressons à des systèmes sur lesquels la politique de sécurité est implantée au travers d'un contrôle d'accès discrétionnaire classique. En outre, l'absence de droit dans une matrice de contrôle d'accès équivalant *de facto* à une interdiction, nous considérons que cette absence de droit constitue *de jure* une interdiction.

A titre d'exemple, considérons, sur un tel système, 3 fichiers[5] `f1`, `f2` et `f3` et deux utilisateurs `U1` et `U2` possédant à l'initialisation des droits sur ces fichiers définis par la matrice d'accès suivante :

	U1	U2
f1	r	
f2	w	r
f3		w

L'absence de droit équivalant à une interdiction, ces droits traduisent une politique qui interdit à `U2` d'accèder au contenu de `f1`. Ainsi, tout flux d'information de `f2` vers `f3` constitue donc *de facto* une violation de la politique, s'il a été précédé d'un flux de `f1` vers `f2`.

Empêcher ou détecter un tel flux illégal permet d'empêcher (schéma de prévention) ou de détecter (schéma de détection) les attaques par délégation grâce auxquelles `U2` tenterait, par effet de bord (comme dans l'attaque de l'impression symbolique) de faire copier le contenu de `f1` dans `f2` par un processus agissant pour le compte de `U1`, pour ensuite accèder lui-même à `f2`.

Au travers de cet exemple, on constate que la légalité de l'accès par un sujet à un objet du système ne doit plus être décidé sur la seule base des droits associés au couple objet-sujet, mais en tenant compte du contenu du fichier. Ainsi, on détecte ou empêche bien toute intrusion se caractérisant par la construction d'un flux d'information non autorisé par la politique.

[5] Le principe reste le même avec tout autre type d'objet : répertoire, socket, etc.

3.3.4 Limitation du nombre de faux positifs

Comme mentionné précédemment dans ce mémoire, un des problèmes majeurs des outils actuels de détection d'intrusion, est le taux très important de faux positifs générés. Aussi, nous souhaitons n'émettre d'alarme qu'en cas de violation **effective** de la politique, c'est-à-dire en cas de flux illégal effectivement constaté.

En reprenant l'exemple du paragraphe précédent, cela signifie que si, à la suite d'une copie de `f1` dans `f2`, `U2` n'accède jamais à `f2`, aucune alarme ne sera émise. Il en va de même si `f1` est copié dans `f2`, puis `f3` dans `f2`, avant que `U2` n'accède à `f2`. Il en va toujours de même sur l'exemple de l'attaque par impression symbolique, si le fichier imprimé via l'attaque est de toutes façon accessible en lecture à l'attaquant.

Comme déjà mentionné, l'approche proposée ne doit s'appuyer, ni sur la spécification de quelconques signatures d'attaque, ni sur la mise en place d'un quelconque mécanisme d'apprentissage de comportement. L'objectif est bien de proposer une approche orientée par la politique de sécurité : la seule « entrée » du détecteur est la traduction de cette politique en terme de droits d'accès discrétionnaires, une absence de droit équivalant à une interdiction.

3.3.5 Un modèle de contrôle de flux adapté à la détection des attaques par délégation

Nous présentons notre modèle dans ce paragraphe. La figure 3.4 tente d'en donner une représentation.

Objects et opérations

Nous considérons le système d'exploitation comme une collection d'objets. Ces objets peuvent être par exemple des fichiers, des processus ou des descripteurs d'entrée-sortie. Ils contiennent de l'information à laquelle les utilisateurs peuvent tenter d'accèder. Un mécanisme de contrôle leur accorde ou non cet accès. Ce mécanisme est le reflet de la politique de sécurité.

Définition 3.1 *Nous adoptons ici la définition classique de* ***classe d'objet*** *et* ***objet****. Une classe d'objet regroupe des attributs et des méthodes de classe permettant de consulter ou modifier ces attributs. Un objet est une instance de la classe, possédant des instances d'attributs et des méthodes d'objet. Nous notons $o.m$ la méthode m de l'objet o.*

Définition 3.2 *Un* ***objet atomique*** *est un objet qui ne possède pas de méthode permettant de modifier certains attributs sans modifier les autres. Tout objet peut être décomposé en objets atomiques.*

Définition 3.3 *L'****état d'un objet*** *est la valeurs de ses attributs.* ***Information*** *est pour nous synonyme d'état d'un objet.*

Définition 3.4 *Une **opération atomique** est l'exécution d'une méthode d'un objet. Elle se caractèrise donc par l'objet considéré, la méthode invoquée, l'état initial et l'état final de l'objet.*

Définition 3.5 *Une **opération** est un ensemble indivisible d'opérations atomiques identifié dans l'interface du système d'exploitation (abstraction d'un appel système).*

Références et domaines

Le concept de référence modélise l'autorisation d'invoquer une méthode d'objet, étant donné l'état courant de cet objet. Une référence crée un droit sur un objet dans un état donné (c-à-d contenant une information donnée), indépendemment de tout sujet.

Toute opération requiert une ou plusieurs références. Dans le cas le plus simple, l'opération est une opération atomique. C'est le cas de la fermeture d'un descripteur d'entrée-sortie, qui fait appel à la méthode `close` de l'objet descripteur. L'exécution de l'appel système `close(fd)` requiert la référence `close` sur l'objet `fd`. D'autres opérations plus complexes nécessitent plusieurs références : par exemple, `read(fd, buffer, size)` nécessite une référence `read` sur l'objet `fd` et une référence `write` sur l'objet `buffer`[6], engendrant ainsi un flux d'information de l'objet `fd` vers l'objet `buffer`.

Définition 3.6 *Une **référence** sur un objet est la possibilité d'invoquer une méthode de cet objet, au vu de l'état de cet objet.*

Définition 3.7 *Un **domaine** est un ensemble de références dont aucune combinaison n'engendre de flux illégal d'information.*

Toute opération utilisant des références appartenant à un même domaine est par définition légale. *A contrario*, toute opération utilisant des références de plusieurs domaines est illégale et doit être empêchée ou détectée (c-à-d donner lieu à une alarme). C'est la règle d'unicité du domaine. Son respect traduit la légalité d'une opération. C'est donc cette règle que l'IDS vérifiera, pour chaque opération.

La construction initiale des domaines est une traduction de la politique de sécurité. Dans le cas où celle-ci est mise en œuvre par un contrôle d'accès discrétionnaire, la construction des domaines consiste à identifier, utilisateur par utilisateur (en dehors de l'administrateur), l'ensemble de tous les flux qui peuvent être engendrés au travers de ses actions. Ce sont simplement les flux de tout fichier accessible en lecture, vers tout fichier accessible en écriture. Tous les sous-ensembles sont éliminés[7].

[6] Une page mémoire possède la même interface qu'un fichier. Elle est donc représentable par la même classe d'objet.

[7] En effet, les éliminer ne change en rien la règle d'unicité des domaines. Il faut en outre noter que dans la pratique, même si c'est possible, il est extrémement rare qu'un utilisateur possède strictement un sous-ensemble des droits d'un autre. Chaque utilisateur dispose en particulier de droits spécifiques sur son propre compte. Ces droits ne sont pas partagés avec d'autres, rendant les sous-ensembles de droits disjoints.

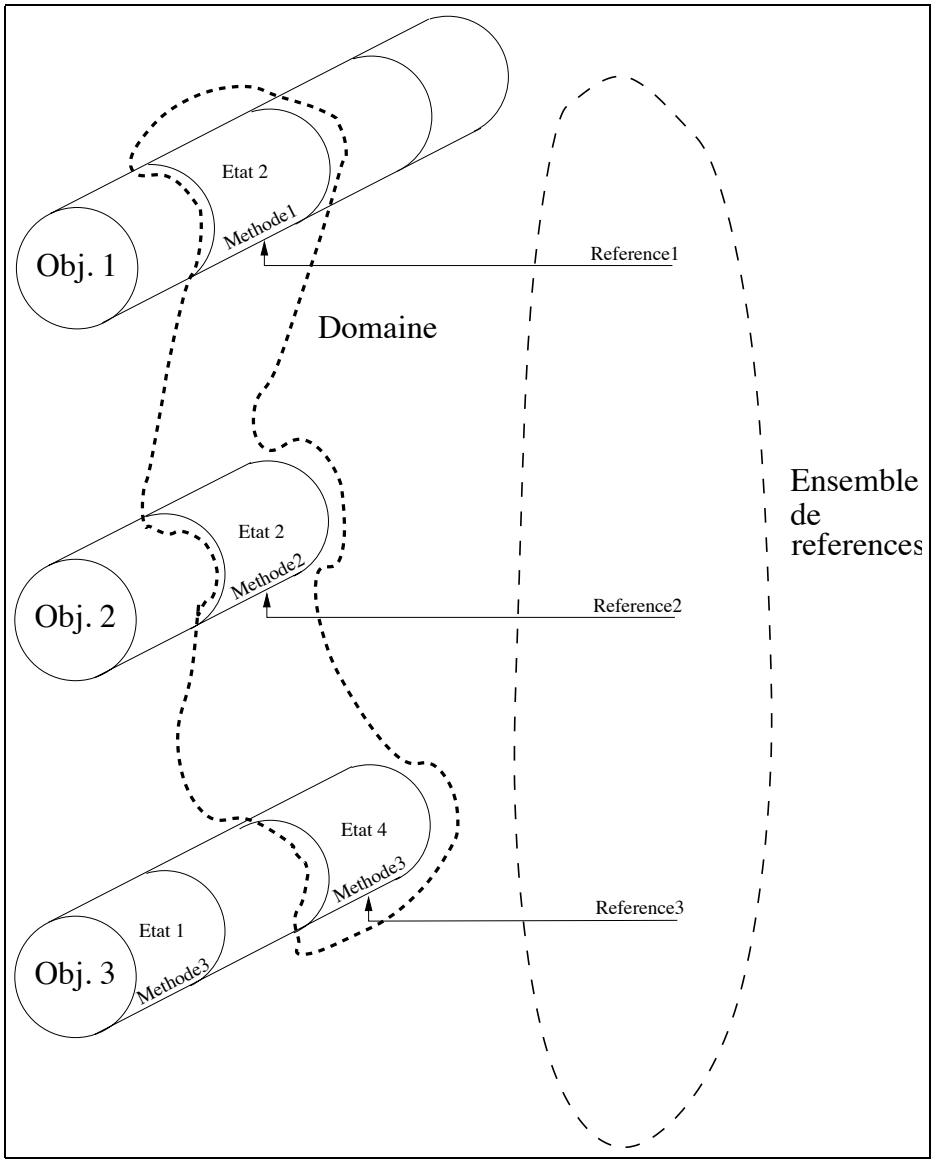


FIG. 3.4 – Objets, méthodes, domaines, références

Les ensembles restant forment autant de domaines.

Ainsi, si l'on considère les fichiers, les groupes et la matrice résultante de le figure 3.5 :

- les flux de `f1`, `f3`, `f4`, `f5` et `f6`, d'une part vers `f4` et `f6`, d'autre part vers `f3`, `f4` et `f6`, sont respectivement possibles via les actions de `U1` et `U3`;
- les flux de `f2`, `f3`, `f4`, `f5` et `f6` vers `f2`, `f5` et `f6` sont possibles via les actions de `U2`.

Les droits de `U1` étant un sous-ensemble de ceux de `U3`, les deux domaines A et B créés sont :

1. $R_A f1.read$, $R_A f3.read$, $R_A f3.write$, $R_A f4.read$, $R_A f4.write$, $R_A f5.read$, $R_A f6.read$, $R_A f6.write$
2. $R_B f2.read$, $R_B f2.write$, $R_B f3.read$, $R_B f4.read$, $R_B f5.read$, $R_B f5.write$, $R_B f6.read$, $R_B f6.write$

On peut associer à chaque object du système les références qui lui correspondent. Pour `f1` : $R_A f1.read$; pour `f2` : $R_B f2.read$, $R_B f2.write$; pour `f3` : $R_A f3.read$, $R_A f3.write$, $R_B f3.read$; etc. A l'initialisation, bien évidemment, ces listes de références par objet sont parfaitement similaires aux droits discrétionnaires d'une ligne de la matrice. Ensuite, en fonction des actions des utilisateurs, cette situation évolue, par le biais des flux de références, présentés dans le paragraphe suivant.

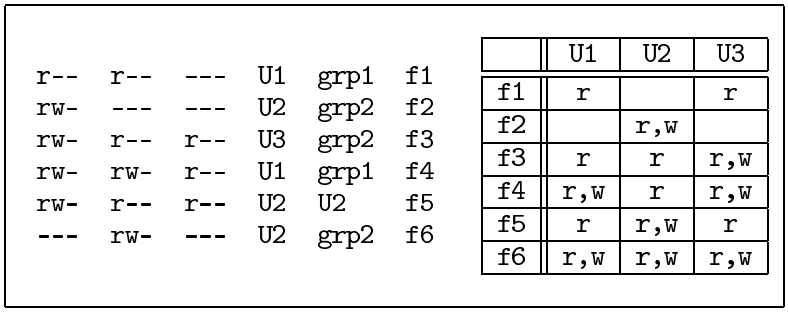
Note : dans la pratique, on gère aussi un « objet virtuel » OV_i par utilisateur i. A chaque objet virtuel, sont associées les références $R_j OV_i.read$ et $R_j OV_i.write$, j étant le domaine des flux possibles pour i. Cette objet virtuel permet, par le mécanisme de flux de référence, de créer les références de « l'objet terminal » (le tty) lors de la connexion d'un utilisateur, qui peut alors saisir et afficher de l'information dans son terminal : cette saisie (resp. affichage) est légale si l'information saisie (resp. affichée) va vers (resp. vient de) un objet du méme domaine (règle d'unicité). Un mécanisme du même ordre est utilisé pour les connexion sans authentification, comme les connexions HTTP.

Dépendance causale et flux de références

Lorsqu'une opération produit un état final d'objet et qu'une autre opération utilise cet état en tant qu'état initial, il existe une dépendance causale entre ces deux opérations qui forment alors une opération composée.

Définition 3.8 *L'ordre causal entre opérations atomiques est défini d'après Lamport [Lam78] :*

- *Etant donné un objet O et deux opérations atomiques Ω_1 et Ω_2 portant sur cet objet, Ω_2 dépend causalement de Ω_1 (notation : $\Omega_1 \rightarrow \Omega_2$) si l'état initial de O au début Ω_2 est l'état final de O à l'issue de Ω_1.*
- *Etant donné un objet O et trois opérations atomiques Ω_1, Ω_2 et Ω_3 portant sur cet objet, si $\Omega_1 \rightarrow \Omega_2$ et $\Omega_2 \rightarrow \Omega_3$, alors $\Omega_1 \rightarrow \Omega_3$.*



```
r--  r--  ---  U1  grp1  f1
rw-  ---  ---  U2  grp2  f2
rw-  r--  r--  U3  grp2  f3
rw-  rw-  r--  U1  grp1  f4
rw-  r--  r--  U2  U2    f5
---  rw-  ---  U2  grp2  f6
```

	U1	U2	U3
f1	r		r
f2		r,w	
f3	r	r	r,w
f4	r,w	r	r,w
f5	r	r,w	r
f6	r,w	r,w	r,w

FIG. 3.5 – Avec les droits indiqués sur la partie gauche de la figure, et en supposant que `U3` soit le seul membre du groupe `grp1` et que `U1`, `U2` et `U3` soient tous membres de `grp2`, la matrice d'accès résultante est donnée sur la partie droite de la figure.

L'ordre causal sur les opérations résulte directement de l'ordre causal des opérations atomiques qui les composent.

Pour ne pas créer de flux illégal, toute opération composée, au même titre qu'une opération simple, doit utiliser des références appartenant au même domaine. En outre, les références associées aux objects destination doivent évoluer afin de tenir compte du flux d'information qui s'est produit. C'est ce qu'on appelle le flux de références.

En reprenant l'exemple du paragraphe précédent, voyons comment évoluent les références associées à `f6` si `U2` y copie successivement `f2`, puis `f3` (initialement, les références associées à `f6` sont : $R_A f6.read$, $R_A f6.write$, $R_B f6.read$, $R_B f6.write$) :

1. copie de `f2` dans `f6` :
   - l'opération est légale car elle nécessite deux références du même domaine B : $R_B f2.read$ et $R_B f6.write$;
   - les références pour `f2` dans le domaine B (seul domaine dans lequel des opération sur `f2` sont possibles) sont transférées à `f6`, dont les références associées deviennent : $R_B f6.read$, $R_B f6.write$;
   - en conséquence, ni `U1`, ni `U3` ne peuvent plus légalement accéder au contenu de `f6`. En effet, pour le faire, ils devraient transférer le contenu de `f6` dans un objet mémoire du domaine A, violant par la même la règle d'unicité des domaines.
2. copie de `f3` dans `f6` :
   - l'opération est légale car elle nécessite deux références du même domaine (en l'occurrence ce flux est possible dans A comme dans B) ;
   - les références pour `f3` dans les domaines A et B sont transférées à `f6`, dont les références associées deviennent : $R_A f6.read$, $R_A f6.write$, $R_B f6.read$;
   - en conséquence, l'accès à l'information contenue dans `f6` redevient possible pour `U1` et `U3`. On constate en outre qu'une des références initialement associée à `f6`, $R_B f6.write$ ne l'est plus. En effet, le contenu de `f3` n'est pas modifiable et, en conséquence, celui, identique, de `f6` ne doit pas l'être.

Dans cet exemple, aucun nouvel objet n'est créé. Il va de soi que si c'était le cas, de nouvelles références devrait être associée à cette objet nouvellement créé et que le mécanisme de flux de références s'appliquerait à l'identique.

3.3.6 Exemple d'implantation d'une politique de sécurité simple par le biais du contrôle des flux de référence

Reprenons la politique simple qui introduit le paragraphe 3.3.1 :
- chaque utilisateur est authentifié par un mot de passe,
- chaque utilisateur dispose d'un accès illimité aux fichiers dont il est le propriétaire,
- chaque utilisateur peut modifier lui-même son mot de passe (ce qui implique la lecture et l'écriture dans ce fichier de mots de passe),
- l'accès au fichier contenant les mots de passe est interdit à tous sauf à l'administrateur.

Prenons en compte 3 classes d'objets : les fichiers, les descripteurs d'entrée-sortie et les processus. Les méthodes de classe correspondantes sont les suivantes :
- fichier : ouverture en lecture, ouverture en écriture, ouverture en lecture et écriture (notation : `open-read()`, `open-write()`, `open-read-write()`)
- descripteur d'entrée-sortie : lecture, écriture, fermeture (notation : `read()`, `write()`, `close()`);
- processus : destruction, suspension, reveil.

Pour implanter la politique, il faut considérer deux objets : le fichier des mots de passe (`/etc/shadow`), la commande de modification du mot de passe (`/bin/passwd`).

Considérons que pour chaque utilisateur, un domaine est créé qui contient, d'une part des références sur les objets de l'utilisateur, d'autre part des références sur les commandes qu'il peut utiliser, dont `/bin/passwd`.

Dans un système classique, la commande `/bin/passwd` est par construction un programme privilégié et `/etc/shadow` est inaccessible aux utilisateurs. De même, dans notre cas, par construction, ces deux objets ont été créés dans le domaine de modification des mots de passe. Les objets du domaine de modification des mots de passe sont `/etc/shadow`, `/bin/passwd` et la console (`stdin`, `stdout`).

Le domaine de modification des mots de passe est constitué des trois méthodes d'ouverture de `/etc/shadow`, de la méthode de lecture de `stdin` et de celle d'écriture de `stdout`.

Aucun autre domaine que celui des modification des mots de passe ne doit contenir de référence d'ouverture de `/etc/shadow`. On assure cette propriété par construction des domaines.

Pour modifier son mot de passe, un utilisateur a besoin d'une référence pour ouvrir `/etc/shadow` en écriture. Comme nous venons de le dire, initialement l'utilisateur n'en dispose pas. Par contre, il dispose d'une référence vers `/bin/passwd`, qu'il peut donc exécuter. L'exécution de `/bin/passwd` est causalement dépendante de la création de `/bin/passwd` et par conséquent elle hérite de références du domaine de modification des mots de passe. Désormais les références d'accès à `/etc/shadow` sont disponibles dans le processus d'exécution de `/bin/passwd`. Le mot de passe est modifiable.

Cependant, toute écriture dans `/etc/shadow` suppose que le contenu écrit puisse être lu ou produit dans le domaine de modification des mots de passe. En effet, l'exécution d'une opération utilisant des références de plusieurs domaines est illégale. Or le seul domaine permettant d'écrire dans `/etc/shadow` est celui de modification des mots de passe. Il est donc illégal de copier le contenu d'un autre fichier dans `/etc/shadow`, de même qu'il est illégal de copier `/etc/shadow` dans un autre fichier.

Pour terminer, on observe qu'avec une architecture de type Unix telle que celle de cet exemple, le modèle implique dans les faits que `/bin/passwd` soit un programme privilégié, car `/etc/shadow` n'est pas un objet atomique. En décomposant `/etc/shadow` en autant d'objets atomiques que de mots de passe, chaque utilisateur pourrait disposer directement des références permettant de modifier son propre mot de passe. Notons que cette « atomisation » est possible sur un système Unix classique, mais dans ce cas, une attaque par délégation serait possible (par exemple par une attaque du type de celle de l'impression symbolique) ; elle permettrait à un utilisateur de lire ou modifier le mot de passe d'un autre. Dans notre cas, une telle attaque est impossible.

En outre, un système Unix est vulnérable à toute attaque qui corrompt la commande `/bin/passwd` (virus, cheval de Troie). Dans notre cas, si jamais une telle corruption se produit et que l'attaquant tente, par exemple, de copier le contenu de `/etc/shadow` sur son propre compte pour déchiffrer les mots de passe, une telle opération nécessiterait des références d'au moins deux domaines, ce qui est par définition illégal.

3.3.7 Exemple de détection d'attaque

Nous présentons ici une attaque de type « *User-to-Root* » [Ken99], permettant à un utilisateur qui dispose d'un compte sur une machine d'acquérir des privilèges de niveau administrateur. L'attaque exploite une vulnérabilité publiée en novembre 2001 [CER01] et visait à l'origine une faille du démon telnet, qui a été corrigée. Par la suite, il a été découvert que cette même attaque pouvait s'appliquer à `open-ssh`, ce qui montre les limites d'une approche de type « *patch-and-pray* », consistant, au coup par coup, à corriger les problèmes qui se présentent, sans s'attaquer à la cause source de ces problèmes. Bien que déjà vieille de 18 mois au moment où ce mémoire est écrit, cette attaque n'avait pas été publiée et nous était inconnue au moment de la conception du modèle de détection par flux de références. Ceci illustre la capacité de l'approche à détecter de nouvelles attaques.

	attaquant	démon ssh
1	création et installation de *libroot.so*	
2	affectation de la variable *LD_PRELOAD* à la valeur *libroot.so*	
3	Démarrage de la session ssh	
4		exécution de */bin/login*
5		chargement de *libroot.so*
6		connexion en tant qu'utilisateur
7		exécution de *setuid(0)*
8		exécution de */bin/sh*
9	shell root disponible	

FIG. 3.6 – Attaque contre le démon ssh

Présentation de l'attaque

L'attaque permet à un utilisateur malicieux de se connecter en tant qu'administrateur sans passer par la procédure d'authentification. En effet, le démon `open-ssh` permet aux utilisateurs de définir des variables d'environnement qui seront positionnées lors de la connexion. En affectant la variable `LD_PRELOAD` à la valeur `libroot.so`, l'utilisateur entraîne le chargement de cette bibliothèque à sa prochaine connexion. Cette bibliothèque, développée par l'utilisateur lui-même, contient une version de l'appel système `setuid` qui fixe systématiquement l'identifiant de l'utilisateur (`uid`) à 0, qui est la valeur de l'identifiant de l'administrateur.

Lors de la connexion suivante, la bibliothèque est donc chargée à l'exécution du processus de connexion. Après authentification (par le mot de passe utilisateur), ce processus exécute `setuid` pour se transformer en processus utilisateur. Or, la bibliothèque `libroot.so` contient la nouvelle version de `setuid`. Etant chargée en dernier, cette version est donc prioritaire sur le `setuid` du système. L'`uid` est donc fixé à 0. Ensuite, le processus de connexion lance un shell pour cet utilisateur. C'est donc un shell `root`.

Les étapes de l'attaque sont présentées par la figure 3.6. Il faut noter, comme mentionné en 3.3.1, que chacune de ces étapes est légale en elle-même. C'est leur enchaînement qui produit un effet indésirable. Cette enchaînement (scénario d'attaque) est inconnu du détecteur.

Détection de l'attaque

L'étape 5 dépend causalement de l'étape 1 puisqu'elle lit un fichier produit à cette étape. Elle hérite donc des références de l'utilisateur. L'utilisation de la

bibliothèque va nécessiter les références de l'utilisateur.

L'étape 7 constitue un appel à cette bibliothèque. Elle a donc besoin de ces références. En revanche, `setuid(0)` (`setuid` est ici la version système) est une opération nécessitant des références non disponibles pour un utilisateur ordinaire, mais héritées du processus de connexion. L'étape 7 nécessite donc des références de deux domaines, ce qui viole la règle d'unicité du domaine. Une opération illégale est ainsi détectée.

3.3.8 Réaction à une opération illégale : schémas de détection et de prévention

Une opération illégale est le symptôme d'une tentative d'attaque. Dés lors, deux stratégies sont envisageables :

- schéma de détection : on laisse l'opération s'exécuter et on émet une alerte ;
- schéma de prévention : on interdit l'exécution de l'opération. Le mécanisme proposé place alors une barrière entre l'attaque et ses conséquences éventuelles.

La maquette que nous avons développé suit la première statégie. Nous nous placons donc clairement dans le domaine de la détection des intrusions. Pour autant, le mécanisme permet très simplement de se placer dans le schéma de prévention : il suffit d'empêcher les appels systèmes violant la politique. Il ne nous appartient pas de décider le schéma qui doit être suivi. C'est un point qui entre aussi dans la politique de sécurité. L'important est de noter que le mécanisme proposé permet l'un et l'autre.

3.3.9 Avantages et limites

Le modèle de détection ne fait aucune hypothèse en ce qui concerne le scénario menant à une opération illégale. Une tentative de cumuler illégalement les effets de plusieurs opérations légales aboutira à une opération utilisant des références de plusieurs domaines, quelque soit la séquence de ces opérations. De nouvelles attaques peuvent donc être détectées.

Toute attaque menant à une violation de la règle d'unicité des domaines est détectable. Cette attaque peut être locale, comme pour tous les exemples présentés. Elle peut aussi être distante. Ainsi, si un attaquant distant réussi, par *buffer overflow*, à obtenir un shell root sur un serveur web, sera jugé illégal tout accès via ce shell root à un fichier non accéssible par le biais d'un accès web normal. En effet, un tel accès produirait un flux d'un fichier d'un certain domaine, vers une socket de domaine `nobody`. La règle d'unicité n'étant pas respectée, ce flux est bien illégal.

Des scénarios qui pourrait paraître suspects, mais qui restent pourtant légaux ne sont pas détectés en tant qu'attaques, ce qui limite le taux de faux positifs. Il en est ainsi si un utilisateur exploite l'attaque de l'impression symbolique pour imprimer un fichier auquel il a par ailleurs légalement accès. De même, en reprenant l'exemple

```
char buf1{[}10000{]}, buf2{[}10000{]};
int fd1, fd2, size;
fd1=open("/etc/shadow", O\_RDONLY);
fd2=open("/home/user1/copie", O\_WRONLY);
size=read(fd1, buf1, 10000);
memcpy(buf2, buf1, size);
write(fd2, buf2, size);
close(fd1);
close(fd2);
```

FIG. 3.7 – Attaque non détectable par le mécanisme présenté.

ci-dessus, l'accès par l'attaquant via son shell root à `/www/index.html` est jugé légal, puisque de toutes façon l'accès à `/www/index.html` est possible publiquement via une requête HTTP.

Etant de niveau de granularité objet, ce modèle ne peut pas détecter des attaques ne reposant pas effectivement sur un accès illégal aux dits objets. Ainsi, l'attaque aboutissant à l'exécution du programme présenté par la figure 3.7, ne sera détectée que si la zone de données du processus exécutant ce programme est un unique objet. En effet, dans le cas contraire, en contrôlant simplement les opérations système exécutées par ce programme sans une connaissance précise de son fonctionnement, il n'est pas possible de déduire que l'opération d'écriture dans *fd2* est la conséquence causale de la lecture sur *fd1*. Pour rendre cette attaque détectable, il faut donc adopter une granularité telle que la zone de donnée du processus soit un objet atomique. C'est ce que nous avons retenu dans notre implémentation.

Conclusion

Le modèle que nous proposons convient à la détection des violations de la politique de contrôle d'accès en raisonnant au niveau du flux d'information produit par les accès, mais n'est pas à même de détecter des violations d'une politique par exploitation d'un canal caché. Le modèle est facilement implémentable car il repose uniquement sur le contrôle des opérations système exécutées. Il fournit un mécanisme de détection ou de prévention des intrusions lors de l'exécution.

Cette approche nous paraît intéressante car elle permet d'intégrer le mécanisme anti-intrusions au cœur du système d'exploitation. Ce n'est plus une sur-couche plus ou moins adaptée. En outre, la statégie adoptée lorsqu'un ensemble de droits permet d'exécuter une opération interdite permet de faire de la détection d'intrusions (cas de l'émission d'une alerte) ou de la prévention d'intrusions, puisqu'on peut interdire l'exécution de l'opération. Le mécanisme proposé permet ainsi de placer une barrière entre l'attaque et ses conséquences. Cette approche se distingue ainsi de la plupart des mécanismes aujourd'hui proposés par les IDS.

Actuellement, nous disposons d'une maquette sous Linux qui a permis de valider l'approche. Par exemple, l'attaque contre `open-ssh` présentée dans le paragraphe 3.3.7, inconnue de nous lors des développements, a été parfaitement détectée dès lors que nous l'avons mise en œuvre. Nous avons implanté le mécanisme sur le serveur de messagerie et le serveur web de l'équipe SSIR (une douzaine d'utilisateurs). Les résultats obtenus sont très intéressants : toutes les attaques détectables sont détectées, seules les violations effectives de la politique donnent lieu à alarme, le taux de faux positif est très faible. Un article présentant ces résultats vient d'être soumis à RAID 2003 [ZMB03a].

L'ensemble de ce travail a été réalisé dans le cadre de la thèse de Jacob Zimmermann et en collaboration avec Christophe Bidan. Une version préliminaire du modèle de détection a fait l'objet d'une communication à RAID en octobre 2002 [ZMB02]. Une version améliorée du modèle, plus claire et plus élégante, vient d'être soumise à ESORICS 2003 [ZMB03b].

Chapitre 4

Perspectives de recherche

Ce mémoire constitue une synthèse du travail de recherche que nous avons mené dans le domaine de la détection d'intrusions. Son organisation répond à la conviction que les études que nous avons successivement menées ont toutes contribuées à forger : la relative incapacité des IDS actuels à remplir leur office est liée à leur totale ou quasi-totale absence de prise en compte de l'environnement dans lequel ils sont placés. Aussi, après avoir posé la problématique de ce domaine, nous avons présenté les grandes caractéristiques des systèmes de détection d'intrusions, positionné l'ensemble de nos travaux relativement à ces caractéristiques, puis davantage détaillé deux études, dont l'objectif commun est l'amélioration des performances de détection par la prise en compte des caractéristiques du système d'information surveillé, au niveau sonde et au niveau manager.

Malgré les résultats obtenus, l'objectif fixé il y a une dizaine d'année, « détecter de manière efficace les attaques contre les systèmes d'information », est bien loin d'être atteint. Il reste pourtant toujours d'actualité selon nous, si bien qu'il continue de structurer notre projet de recherche. Les études menées par le passé contribuent, par leurs réussites, mais aussi par leurs échecs, à orienter les travaux que nous souhaitons maintenant conduire. Ceux-ci s'inscrivent dans la double nécessité d'améliorer à la fois les sondes, qui produisent des alertes, et les managers, qui consomment ces alertes et, éventuellement, en produisent de nouvelles. En outre, dans tous les cas, l'évaluation des mécanismes proposés doit être rigoureuse. Nous concluons ce mémoire par une brève description de quelques travaux futurs, que nous comptons mener sur ces 3 plans.

4.1 Amélioration des sondes de détection

Nous continuerons à investiguer à la fois l'approche comportementale et l'approche par scénario. En outre, quelque soit l'approche, nous continuerons à améliorer la vision que les sondes possèdent de leur environnement.

4.1.1 Approche comportementale

Un premier axe consistera à renforcer le travail fait dans le cadre de l'apprentissage de comportement (modèle à mélange de gaussienne utilisé dans la thèse de Zakia Marrakchi).

Un premier travail, déjà entamé récemment dans le cadre de la thèse de Ricardo Puttini, propose une méthode de détection des attaques de type déni de service[1]. Les approches actuelles proposent d'apprendre les seuils de certaines variables (par exemple de volume de trafic ou de nombre de requête) au delà desquels on suspecte une attaque par inondation. Cette approche provoque bien sûr des faux positifs lorsque le seuil est dépassé à cause d'une augmentation « légitime » du trafic ou du nombre de requête. Nous pensons qu'en prenant en compte les corrélations entre les variables, il pourrait être possible de mieux discriminer une inondation d'une augmentation légitime. Le modèle à mélange de gaussienne est adapté à cet objectif. Une implémentation est en cours de réalisation pour valider cette idée, dans le cadre d'un environnement de type réseau ad hoc. Les variables surveillées sont celles de la MIB (*Management Information Base*). Si les résultats sont concluants, nous envisagerons de coupler le détecteur aux travaux qui permettent de remonter, routeur par routeur, aux sources de l'attaque, afin de la bloquer.

Une autre évolution du travail de Zakia Marrakchi consistera à revenir sur la nature arborescente de la représentation du comportement. Nous envisageons, avec Bernard Vivinis (enseignant-chercheur de l'équipe SSIR de Supélec), de passer à une représentation sous forme d'automate, plus compacte et *a priori* permettant de représenter des comportements plus complexes. Dès lors, se pose le problème, non trivial, de l'apprentissage d'automate. Ce travail n'a pas encore débuté.

Enfin, l'application des propositions de la thèse de Zakia Marrakchi à d'autres environnements que CORBA est aussi envisagée. Ainsi, dans le cadre de la thèse de Elvis Tombini, nous allons avoir besoin de modéliser l'usage normal d'un serveur web. Il est raisonnable d'envisager pour cela d'utiliser des arbres ou des automates de comportement, couplés à un modèle de mélange de gaussienne pour prendre en compte la répartition des paramètres des requête HTTP. Un point important ici sera la prise en compte des paramètres non numériques.

Un deuxième axe réside dans la poursuite de travaux dans lesquels le comportement de référence est construit *a priori*.

Bien évidemment, entre dans cet axe les développements que nous envisageons pour les travaux de Jacob Zimmermann autour de la détection d'intrusions orientée politique.

En premier lieu, comme toute approche comportemental, ce travail pêche par la pauvreté du diagnostic associé aux alertes. Ainsi, si nous sommes capable de vérifier

[1] Dans le passé, nous ne nous sommes que très peu intéressé à ce type d'attaque, pourtant important. Lors de mon séjour sabbatique à l'Université de Davis, j'ai entamé, avec Matt Bishop, un travail sur la détection des dénis de service. Ce travail a ensuite été poursuivi par des étudiants de 3ème année à Supélec. Malheureusement, depuis lors, il n'a plus évolué, par manque de ressources.

la légalité de tout flux d'information sur la machine, nous ne savons pas a l'heure actuelle dire comment un flux illégal constaté a été généré. La prochaine étape de notre travail consistera donc à proposer une méthode de diagnostic applicable à toute alerte émise. Dans l'idéal, on devrait être capable de fournir le scénario d'attaque qui a conduit à la violation constatée de la politique. Ce travail devrait être accompli en collaboration avec George Mohay, de la *Queensland University of Technology* (Brisbane, Australie), où Jacob Zimmermann devrait effectuer un séjour postdoctoral, si la bourse qu'il a demandé lui est accordé.

La version actuelle du modèle de détection proposé est centralisée. Un démon est installé sur une machine et ce démon vérifie la légalité des flux d'information sur cette machine. Une évolution « naturelle » de ce modèle réside dans sa distribution. L'objectif sera cette fois de vérifier la légalité des flux d'information entre machines. Bien évidemment, à cette fin, une politique distribuée devra être spécifiée. Ce sera la première étape de ce travail.

Nous souhaitons également lancer, toujours sur l'axe « référence de comportement fixée *a priori* », un travail de recherche sur la détection d'intrusion orientée spécification. Cette fois, le comportement de référence n'est pas fixé par une politique de sécurité, mais par une spécification logiciel (dans le sens attribué au terme en génie logiciel). L'idée est en fait de s'inspirer de la diversification fonctionnelle, approche utilisée avec succès dans le domaine de la sûreté de fonctionnement. Sur la base d'une même spécification (par exemple, celle du protocole HTTP) plusieurs logiciels sont développés, installés et utilisés en parallèle. Il est alors possible, en cours d'utilisation, d'observer les écarts de comportement entre ces différents logiciels (chacun constituant en quelque sorte la référence des autres). En cas d'écart constaté, on suspecte une intrusion. Cette étude est un des travaux proposés dans une réponse que nous venons de faire à l'appel à proposition lancé en avril 2003 par le ministère de la recherche dans le cadre de l'action concertée incitative « sécurité informatique ». Si cette proposition est acceptée, le travail sera réalisé en collaboration avec Eric Totel, enseignant-chercheur de l'équipe SSIR de Supélec.

4.1.2 Approche par scénario

Comme le montre le tableau 2.2 de la page 22, nous avons longtemps privilégié l'approche par scénario dans nos travaux, les défauts intrinsèques de l'approche comportemental nous semblant alors rédhibitoires. Aujourd'hui, tout en n'étant pas de ceux qui pensent que l'approche par scénario n'est pas réaliste, nous souhaitons, avant de poursuivre éventuellement nos travaux sur cet axe, consacrer une étude à la redéfinition du concept même de scénarios : que doivent-ils être, que peuvent-ils être, que ne peuvent-ils pas être. En d'autres termes, nous ne sommes plus certains aujourd'hui (mais sans être non plus certains du contraire) que le plus pertinent soit d'exprimer sous forme de signatures les phénomènes que l'on veut détecter.

Pour augmenter nos connaissances sur les techniques réellement utilisées actuellement par la communauté des « hackers », nous nous sommes impliqué dans une réponse au dernier appel à projet du RNRT (printemps 2003). Le projet consiste

à définir et mettre en place sur nos réseaux des mécanismes de leurre (type *honey pot*). Ces mécanismes permettront de collecter des informations sur des activités *a priori* intrusives, puisque seuls des attaquants devraient se mettre en rapport avec les machines sur lesquelles seront installés nos leurres, ces machine n'offrant aucun service opérationnel. A la lumière des données ainsi collectées, nous espérons être en mesure de définir les mécanismes de détection aujourd'hui les plus adaptés. Si de ces travaux il ressort que l'approche par scénario semble la meilleure dans tel ou tel contexte, nous alimenterons nos travaux antérieurs ($G^A_S SA_T A$, $G^N G$, ADeLe) avec des signatures correspondant à ces contextes.

4.1.3 Prise en compte de l'environnement par les sondes

Nous avons identifié, avec M2D2, les informations dont il faut tenir compte dans le processus de corrélation des alertes. Ces informations portent sur les caractéristiques du système surveillé : topologie, logiciels installés, failles connues, politique en vigueur. Elles permettent de limiter le nombre d'alerte et d'enrichir le diagnostic.

Une partie de ces informations, sinon toutes, pourrait aussi être mise à profit dans les sondes[2]. Une évolution intéressante du travail réalisé par Benjamin Morin autour de M2D2 serait donc de distribué à chaque sonde les informations lui étant utiles. Dés lors, le travail de limitation du nombre d'alertes (par exemple d'élimination des faux positifs) pourrait commencer au plus tôt, dès la sonde. Appliquée à des sondes de type snort, cette idée revient à enrichir les règles de matching par des éléments de contexte.

4.2 Amélioration des fonctions du manager

Nos travaux futurs sont bien évidemment liés, dans le cadre de l'amélioration des fonctions du manager, au modèle M2D2 présenté dans la section 3.2. (Des travaux sur l'exploitation de M2D2 sont actuellement en cours : exploitation par chroniques pour exprimer des règles de corrélation explicite, exploitation par un algorithme de clustering s'appuyant sur des arbres de concepts pour exprimer des règles de corrélation implicite. Ces travaux sont réalisés dans le cadre de la thèse de Benjamin Morin.)

M2D2 doit être enrichi, car il n'intègre actuellement qu'une partie des connaissances utiles à la corrélation. Il lui manque des informations sur les signatures d'attaque (couplage possible avec le langage ADeLe), les types de machines et les circonstances dans lesquelles ils peuvent donner lieu à alerte(s) en l'absence d'attaque (faux positif connus), le comportement des sondes face à des attaques particulières (par exemple émission récurrente d'alertes, phénomène qui pourrait également être décrit en ADeLe), la politique de sécurité. L'ajout de ces informations constitue bien évidemment la prochaine étape de travail. Ces informations permettront de nouvelles formes de corrélation.

[2]Les flux de références sont une manière de prendre en compte la politique de sécurité pour un type de sonde particulier.

4.3 Evaluation des IDS

Les évaluations qui sont réalisées aujourd'hui (y compris les notres) ne sont pas de bonne qualité : les mécanismes proposés sont confrontés à trop peu d'attaques ; les activités non intrusives au milieu desquelles ces attaques sont noyées ne constituent que des cas particuliers et sont mal caractérisées ; enfin, et c'est un problème majeur pour notre activité de recherche scientifique, les données ne sont généralement pas accessibles et les expérimentations faites pas reproductibles.

Nous avons mentionné ci-dessus notre implication dans un projet visant à mettre en place des leurres afin de collecter de l'information sur les attaques. Un autre apport essentiel de cette étude sera de fournir des logs de tous types (réseau, système, application, IDS) qui permettront de constituer des *benchmarks* pour l'évaluation de performances des IDS. A cette fin, un travail important de caractérisation, de documentation et de mise à disposition des logs devra être fait.

Bibliographie

[Abr96] J.R. Abrial. *The B Book : Assigning programs to meanings.* Cambridge University Press, 1996.

[ACP+02] Patrick Albers, Olivier Camp, Jean-Marc Percher, Bernard Jouga, Ludovic Mé, and Ricardo Puttini. Security in ad hoc networks : a general intrusion detection architecture enhancing trust based approaches. In *Proceedings of the First International Workshop on Wireless Information Systems (WIS-2002)*, April 2002.

[And80] J.P. Anderson. Computer security threat monitoring and surveillance. Technical report, James P. Anderson Company, Fort Washington, Pennsylvania, April 1980.

[Axe99] Stefan Axelsson. Research in intrusion-detection systems : A survey. Technical Report 98-17, Department of Computer Engineering, Chalmers University of Technology, Goteborg, Sweden, August 1999.

[Axe00] Stefan Axelsson. Intrusion detection systems : A taxonomy and survey. Technical Report 99-15, Dept. of Computer Engineering, Chalmers University of Technology, March 2000.

[Bis95] Matt Bishop. A standard audit trail format. Technical report, Department of Computer Science, University of California at Davis, 1995.

[BL73] D. Bell and L. LaPadula. Secure computer systems : mathematical foundations. Technical Report ESD-TR-73-278, The Mitre Corp., 1973.

[BM77] R.S. Boyer and J.S. Moore. A fast string searching algorithm. *Communications of the ACM*, 20 :762–772, 1977.

[CD03] Dave Curry and Hervé Debar. Intrusion detection message exchange format data model and extensible markup language (xml). Internet Draft, draft-ietf-idwg-idmef-xml-10, January 2003.

[CER01] CMU CERT/CC. Vu#40327 : Openssh uselogin option allows remote execution of commands as root. http ://www.kb.cert.org/vuls/id/40327, November 2001.

[CM02] Frédéric Cuppens and Alexandre Miège. Alert correlation in a cooperative intrusion detection framework. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2002.

[CO00] Frédéric Cuppens and Rodolphe Ortalo. Lambda : A language to model a database for detection of attacks. In H. Debar, L. Mé, and S. F. Wu, editors, *Proceedings of the Third International Workshop on the Recent Advances in Intrusion Detection (RAID'2000)*, number 1907 in LNCS, pages 197–216, October 2000.

[Cup01] Frédéric Cuppens. Managing alerts in a multi-intrusion detection environment. In *Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC 2001)*, December 2001.

[DBS92] H. Debar, M. Becker, and D. Siboni. A neural network component for an intrusion detection system. In *Proceedings of the IEEE Symposium of Research in Computer Security and Privacy*, pages 240–250, May 1992.

[DC01] Oliver M. Dain and Robert K. Cunningham. Fusing heterogeneous alert streams into scenarios. In *Proceedings of the Workshop on Data Mining for Security Applications, 8th ACM Conference on Computer and Communication Security*, November 2001.

[DCW+99] Robert Durst, Terrence Champion, Brian Witten, Eric Miller, and Luigi Spagnuolo. Test and evaluation of computer intrusion detection systems. *Communications of the ACM*, 42(7), July 1999.

[DDW00] Hervé Debar, Marc Dacier, and Andreas Wespi. A revised taxonomy for intrusion-detection systems. *Annales des Télécommunications*, 55(7-8), 2000.

[Deb93] Hervé Debar. *Application des réseaux de neurones à la détection d'intrusions sur les systèmes informatiques.* PhD thesis, Université de Paris 6, 1993.

[Den76] Dorothy E. Denning. A lattice model of secure information flow. *ACM*, 19(5), May 1976.

[Den87] D.E. Denning. An intrusion-detection model. *IEEE transaction on Software Engineering*, 13(2) :222–232, 1987.

[Der99] Renaud Deraison. *The Nessus Attack Scripting Language Reference Guide*, September 1999.

[Des01] Yves Deswarte. Les systèmes d'information et leur sécurité. *REE*, (5) :25, mai 2001.

[DGG93] Christophe Dousson, Paul Gaborit, and Malik Ghallab. Situation recognition : Representation and algorithms. *Proceedings of the* 13^{th} *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 166–172, August 1993. Chambéry, France.

[DGKS94] M. Denault, D. Gritzalis, D. Karagiannis, and P. Spirakis. Intrusion detection : approach and performance issues of the securenet system. *Computers & Security*, 13 :495–508, 1994.

[Dou94] Christophe Dousson. *Suivi d'évolutions et reconnaissance de chroniques.* Intelligence artificielle, Université Paul Sabatier, Toulouse, France, September 1994.

[Dou96] Christophe Dousson. Alarm driven supervision for telecommunication networks : II- On-line chronicle recognition. *Annals of Telecommunications*, pages 501–508, October 1996. CNET, France Telecom.

[DW01] Hervé Debar and Andreas Wespi. Aggregation and correlation of intrusion-detection alerts. In W. Lee, L. Mé, and A. Wespi, editors, *Proceedings of the Fourth International Symposium on the Recent Advances in Intrusion Detection (RAID'2001)*, number 2212 in LNCS, pages 85–103, October 2001.

[Equ00] Equipe MIRADOR (Alcatel, ENSTB, ONERA, Supélec). Mirador : a cooperative approach of IDS. Poster présenté au 6éme European Symposium on Research in Computer Security (ESORICS). Toulouse, France, octobre 2000.

[EVK00] Steven T. Eckmann, Giovanni Vigna, and Richard A. Kemmerer. Statl : An attack language for state-based intrusion detection. In *Proceedings of the ACM Workshop on Intrusion Detection*, November 2000.

[FHS97] S. Forrest, S.A. Hofmeyr, and A. Somayaji. Computer immunology. *Communications of the ACM*, 40(10) :88–96, October 1997.

[FKP+99] Rich Feiertag, Cliff Kahn, Phil Porras, Dan Schnackenberg, Stuart Staniford-Chen, and Brian Tung. A common intrusion specification language (cisl). Specification draft, http ://www.gidos.org/drafts/language.txt, June 1999.

[GG01] Christopher W. Geib and Robert P. Goldman. Information modeling for intrusion report aggregation. In *Proceedings of the DARPA Information Survivability Conference and Exposition*, June 2001.

[GG02] Christopher W. Geib and Robert P. Goldman. Requirements for plan recognition in network security systems. In *Proceedings of the Fifth International Symposium on Recent Advances in Intrusion Detection (RAID 2002)*, October 2002.

[GHH+01] R. P. Goldman, W. Heimerdinger, S. A. Harp, C. W. Geib, V. Thomas, and R. L. Carter. Information modeling for intrusion report aggregation. In *Proceedings of the DARPA Information Survivability Conference and Exposition*, June 2001.

[Gol89] D. E. Goldberg. *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley, 1989.

[HDL+90] L. T. Heberlein, G. Dias, K. Levitt, B. Mukherjee, J. Wood, , and D. Wolber. A network security monitor. In *Proceedings ot the 1990 IEEE Computer Society Symposiumon Research in Security and Privacy*, pages 296–304, May 1990.

[Ilg93] K. Ilgun. Ustat : A real-time intrusion detection system for UNIX. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 16–29, 1993.

[JMaM99] Wayne Jansen, Peter Mell, and Tom Karygiannis andDon Marks. Applying mobile agents to intrusion detection and response. Technical

Report IR-6416, National Institute of Standards and Technology, Computer Security Division, October 1999.

[JVL+93] H.S. Javitz, A. Valdes, T.F. Lunt, A. Tamaru, M. Tyson, and J. Lowrance. Next generation intrusion detection expert system (NIDES). Technical Report A016-Rationales, SRI, 1993.

[JW93] G. Jakobson and M. D. Weissman. Alarm correlation. *IEEE Network Magazine*, pages 52–60, 1993.

[Ken99] Kristopher Kendall. A database of computer attacks for the evaluation of intrusion detection systems. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, June 1999.

[KNMH00] Josué Kuri, Gonzalo Navarro, Ludovic Mé, and Laurent Heye. A pattern matching based filter for audit reduction and fast detection of potential intrusions. In H. Debar, L. Mé, and S. F. Wu, editors, *Proceedings of the Third International Workshop on the Recent Advances in Intrusion Detection (RAID'2000)*, number 1907 in LNCS, pages 17–27, October 2000.

[KS94] S. Kumar and E.H. Spafford. A pattern-matching model for misuse intrusion detection. In *Proceedings of the national computer security conference*, pages 11–21, 1994.

[KT02] Christopher Krügel and Thomas Toth. Flexible, mobile agent based intrusion detection for dynamic networks. In *Proceedings of European Wireless 2002*, February 2002.

[KTK01] Christopher Krügel, Thomas Toth, and Clemens Kerer. Decentralized event correlation for intrusion detection. In Springer Verlag, editor, *Proceedings ot the 4th International Conference on Information Security and Cryptology (ICISC 2001)*, Lecture Notes in Computer Science, December 2001.

[Lam78] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7) :558–565, 1978.

[LFG+00] Richard P. Lippmann, David J. Fried, Isaac Graf, Joshua W. Haines, Kristopher R. Kendall, David McClung, Dan Weber, Seth E. Webster, Dan Wyschogrod, Robert K. Cunningham, and Marc A. Zissman. Evaluating intrusion detection systems : The 1998 darpa off-line intrusion detection evaluation. In *Proceedings of the 2000 DARPA Information Survivability Conference and Exposition (DISCEX'00)*, 2000.

[LJ88] T.F. Lunt and R. Jagannathan. A prototype real-time intrusion-detection expert system. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 59–66, 1988.

[LMPT98] Ulf Lindqvist, Douglas Moran, Phillip Porras, and Mabry Tyson. Designing idle : The intrusion data library enterprise. Web proceedings of the First International Workshop on Recent Advances in Intrusion

Detection (RAID'98), http ://www.raid-symposium.org/raid98, September 1998.

[LP99] Ulf Lindqvist and Phillip A. Porras. Detecting computer and network misuse through the production-based expert system toolset (p-best). In IEEE Computer Society Press, editor, *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, May 1999.

[LTL01] Yao-Tsung Lin, Shian-Shyong Tseng, and Shun-Chieh Lin. An intrusion detection model based upon intrusion detection markup language (idml). *Journal of Information Science and Engineering*, 17(6) :899–919, November 2001.

[LWJ98] J.-L. Lin, X.S. Wang, and S. Jajodia. Abstraction-based misuse detection : High-level specifications and adaptable strategies. In *Proceedings of the Eleventh Computer Security Foundations Workshop*, June 1998.

[Mé93] Ludovic Mé. Security audit trail analysis using genetic algorithms. In *Proceedings of the 12th International Conference on Computer Safety, Reliability and Security*, pages 329–340, October 1993.

[Mé94] L. Mé. *Audit de sécurité par algorithmes génétiques*. PhD thesis, Université de Rennes 1 - Numéro d'ordre 1069, 1994.

[Mé98] Ludovic Mé. Gassata, a genetic algorithm as an alternative tool for security audit trails analysis. Web proceedings of the First international workshop on the Recent Advances in Intrusion Detection (RAID'98), http ://www.raid-symposium.org/raid98, September 1998.

[MM01] Cédric Michel and Ludovic Mé. Adele : an attack description language for knowledge-based intrusion detection. In *Proceedings of the 16th International Conference on Information Security (IFIP/SEC 2001)*, pages 353–365, June 2001.

[MMDD02] Benjamin Morin, Ludovic Mé, Hervé Debar, and Mireille Ducassé. M2D2 : A formal data model for IDS alert correlation. In *Proceedings of the 5th International Workshop on the Recent Advances in Intrusion Detection (RAID'2002)*, October 2002.

[MMH01] Ludovic Mé, Cédric Michel, and Laurent Heye. A language for a comprehensive description of attacks. Extended abstract presented at the 2001 IEEE Symposium on Security and Privacy, May 2001.

[MMM+01] Ludovic Mé, Zakia Marrakchi, Cédric Michel, Hervé Debar, and Frédéric Cuppens. La détection d'intrusions : les outils doivent coopérer. *Revue de l'Electricité et de l'Electronique*, 5 :50–55, May 2001.

[MMVM00] Zakia Marrakchi, Ludovic Mé, Bernard Vivinis, and Benjamin Morin. Flexible intrusion detection using variable-length behavior modeling in distributed environment : Application to corba objects. In H. Debar, L. Mé, and S. F. Wu, editors, *Proceedings of the Third International Workshop on the Recent Advances in Intrusion Detection (RAID'2000)*, number 1907 in LNCS, pages 130–144, October 2000.

[Mou97] Abdelaziz Mounji. *Languages and Tools for Rule-Based Distributed Intrusion Detection.* PhD thesis, Universite de Namur, Juillet 1997 1997.

[NC02] Peng Ning and Yun Cui. An intrusion alert correlator based on prerequisites of intrusions. Technical Report TR-2002-01, Department of Computer Science, North Carolina State University, January 2002.

[net03] http ://www.networkintrusion.co.uk/ids.htm, 2003.

[NRC01] Peng Ning, Douglas S. Reeves, and Yun Cui. Correlating alerts using prerequisites of intrusions. Technical Report TR-2001-13, Department of Computer Science, North Carolina State University, December 2001.

[OJC97] Tim Oates, David Jensen, and Paul R Cohen. Automatically acquiring rules for event correlation from event logs. Technical Report 97-14, Dept. of Computer Science, University of Massachusetts/Amherst, 1997.

[Pax98] Vern Paxson. Bro : A system for detecting network intruders in real-time. In *Proc. of the 7th Usenix Security Symposium*, January 1998.

[PD01] Jean-Philippe Pouzol and Mireille Ducassé. From declarative signatures to misuse ids. In W. Lee, L. Mé, and A. Wespi, editors, *Proceedings of the Fourth International Symposium on the Recent Advances in Intrusion Detection (RAID'2001)*, number 2212 in LNCS, pages 1–21, October 2001.

[PMM02] Ricardo Puttini, Zakia Marrakchi, and Ludovic Mé. Bayesian classification model for real-time intrusion detection. In *Proceedings of the 22th International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering (MAXENT'2002)*, August 2002.

[Por92] Phillip Andrew Porras. Stat – a state transition analysis tool for intrusion detection. Master's thesis, University of California, Santa Barbara, 1992.

[PPM+03] Ricardo Puttini, Jean-Marc Percher, Ludovic Mé, Olivier Camp, Bernard Jouga, and Patrick Albers. Un système de détection d'intrusion distribué pour réseaux ad hoc. *Soumis à la revue TSI*, 2003.

[PPMC03] Ricardo Puttini, Jean-Marc Percher, Ludovic Mé, and Olivier Camp. A fully distributed IDS for MANET based on a mobile agent framework. Submitted to *the ACM Special Interest Group on Data Communication (SIGCOMM) Conference*, August 2003.

[RTG94] Pierre Rolin, Laurent Toutain, and Sylvain Gombault. Network security probe. In *Proceedings of the 2nd ACM Conference on Computer and Communication Security*, November 1994.

[Sec98] Secure Networks. *Custom Attack Simulation Language (CASL)*, January 1998.

[Shi00] R. Shirey. Internet security glossary. IETF RFC 2828, 2000.

[Sma88] S.E. Smaha. Haystack : An intrusion detection system. In *Proceedings of the 4th Aerospace Computer Security Application Conference*, pages 37–44, december 1988.

[sna03] http ://www.intersectalliance.com/projects/Snare/index.html, 2003.

[Sob00] Michael Sobirey. Michael sobirey's intrusion detection systems page. http ://www-rnks.informatik.tu-cottbus.de/ sobirey/ids.html, 2000.

[Sun00] Sun Microsystems, Inc. Sunshield basic security module guide, 2000.

[TL00] Steven J. Templeton and Karl Levitt. A requires/provides model for computer attacks. In *Proceedings of the 2000 New Security Paradigms Workshop (NSPW'00)*, pages 31–38, September 2000.

[VEK00] Giovanni Vigna, Steven T. Eckmann, and Richard A. Kemmerer. Attack languages. In *Proceedings of the IEEE Information Survivability Workshop*, October 2000.

[Vig96] Giovanni Vigna. A topological characterization of tcp/ip security. Technical Report TR-96.156, Politecnico di Milano, 1996.

[VL89] H.S. Vaccaro and G.E. Liepins. Detection of anomalous computer session activity. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 1989.

[VS01] Alfonso Valdes and Keith Skinner. Probabilistic alert correlation. In W. Lee, L. Mé, and A. Wespi, editors, *Proceedings of the Fourth International Symposium on the Recent Advances in Intrusion Detection (RAID'2001)*, number 2212 in LNCS, pages 54–68, October 2001.

[WE02] Mark Wood and Mike Erlinger. Intrusion detection message exchange requirements. draft-ietf-idwg-requirements-07, June 2002.

[WZY+99] S.F. Wu, X. Zhao, J. Yuill, P. Chen, M. Erlanger, M.Y. Huang, F. Gong, and F. Wang. Intrusion detection system event correlation mib. internet draft, October 1999.

[ZMB02] Jacob Zimmermann, Ludovic Mé, and Christophe Bidan. Introducing reference flow control for intrusion detection at the os level. In *Proceedings of the 5th International Symposium on the Recent Advances in Intrusion Detection (RAID'2002)*, October 2002.

[ZMB03a] Jacob Zimmermann, Ludovic Mé, and Christophe Bidan. Experimenting a policy-based HIDS based on the reference flow control model. Submitted to *the 6th International Symposium on the Recent Advances in Intrusion Detection (RAID'2003)*, September 2003.

[ZMB03b] Jacob Zimmermann, Ludovic Mé, and Christophe Bidan. An improved reference flow control model for policy-based intrusion detection. Submitted to *the 8th European Symposium on Research in Computer Security (ESORICS)*, October 2003.